Available online at www.sciencedirect.com
ScienceDirect
Procedia Computer Science 1 (2012) 1561–1569
Procedia Computer Science
www.elsevier.com/locate/procedia

International Conference on Computational Science, ICCS 2010

Numerical solution of level dependent quasi-birth-and-death processes

Hendrik Baumann, Werner Sandmann

Clausthal University of Technology, Department of Mathematics, Erzstr. 1, D–38678 Clausthal-Zellerfeld, Germany

Abstract

We consider the numerical computation of stationary distributions for level dependent quasi-birth-and-death processes. An algorithm based on matrix continued fractions is presented and compared to standard solution techniques. Its computational efficiency and numerical stability is demonstrated by numerical examples.

Keywords: Continuous-Time Markov Chains, Block-Tridiagonal Generator Matrices, Level Dependent Quasi-Birth-and-Death Processes, Numerical Solution, Matrix Continued Fractions

1. Introduction

In many applications of multi-dimensional Markov chains, an appropriate numbering of the states yields a chain with block-tridiagonally structured generator or transition probability matrix, that is a quasi-birth-and-death process (QBD). QBDs are well suited for modeling various population processes as the abstract term of a population is interpreted in a wide sense. They apply to a diversity of biological, social, economic and technical processes such as cell growth, biochemical reaction kinetics, epidemics, demographic trends, or queueing systems, amongst others. For complex systems where no explicit expressions can be obtained, there is a great interest in numerical analysis.

If all but boundary blocks are identical, then the QBD is said to be level independent and matrix-geometric techniques [1, 2, 3] provide a powerful machinery. However, level dependent QBDs are often more realistic and while efficient and stable numerical solution techniques are available for level independent QBDs, there are only a few approaches that try to exploit the block structure in the level dependent case [4, 5]. Since their numerical stability is not in general guaranteed one often resorts to numerical solution approaches for general Markov chains [6, 7] without taking advantage of the specific QBD structure.

We consider an algorithm for computing stationary distributions of level dependent QBDs based on matrix continued fractions (MCFs). Our description is focused on the continuous-time case but the algorithm applies analogously to the discrete-time case. Essentially, the algorithm relies on an appropriate generalization of continued fractions to matrices, the convergence of such MCFs if a certain subdominant solution of a second-order vector-matrix recursion exists, and a transformation of the system of equations for the stationary distribution of CTMCs into a first-order recurrence scheme. The relation of MCFs to first-order linear recurrence systems is provided by [8] and the connection

Email addresses: hendrik.baumann@tu-clausthal.de (Hendrik Baumann), werner.sandmann@tu-clausthal.de (Werner Sandmann)

1877-0509
doi:10.1016/j.procs.2010.04.175

to Markov chains as well as its usefulness for analyzing queueing models is exploited in [9, 10]. For the general theory of MCFs, see [11, 12, 13].

In the remainder of this paper, we describe the MCF algorithm in Section 2 and provide numerical examples and comparisons to general (not requiring structured generator matrices) numerical solution approaches for Markov chains in Section 3. Finally, Section 4 concludes the paper and outlines further research directions.

2. Matrix continued fraction algorithm

Let $(X_t)_{t\geq 0}$ be an ergodic level dependent QBD, that is an ergodic CTMC with block-tridiagonally structured generator matrix

$$Q = \begin{pmatrix} Q_{00} & Q_{01} & & & & & \\ Q_{10} & Q_{11} & Q_{12} & & & & \\ & Q_{21} & Q_{22} & Q_{23} & & & \\ & & \ddots & \ddots & \ddots & & \\ & & & Q_{N-1,N-2} & Q_{N-1,N-1} & Q_{N-1,N} \\ & & & & Q_{N,N-1} & Q_{NN} \end{pmatrix} \tag{1}$$

where $Q_{n,n+1} \in \mathbb{R}^{d_n \times d_{n+1}}$, $Q_{nn} \in \mathbb{R}^{d_n \times d_n}$, and $Q_{n,n-1} \in \mathbb{R}^{d_n \times d_{n-1}}$.

Note that in the literature QBDs are often defined as *two-dimensional* Markov chains with block-tridiagonal generator matrices. Though we shall focus on examples with two-dimensional state spaces when presenting numerical results in Section 3, we emphasize that in general we do not need any restrictions with regard to the dimensionality of the state space but only require a block-tridiagonal generator matrix. That is, we define a (continuous-time) QBD as a CTMC $(X_t)_{t\geq 0} = (N_t, C_t))_{t\geq 0}$ with state space $\mathcal{S} = \mathbb{N} \times \mathcal{S}', \mathcal{S}' \subseteq \mathbb{N}^d, d \in \mathbb{N}$ where N_t is called the level component, C_t is the control component and direct state transitions are only possible between states whose level components differ by less than one. Then an appropriate numbering of the states yields the desired block-tridiagonal generator matrix.

The specific structure of Q shall be used to construct a fast and numerically stable algorithm for computing the solution x of $xQ = 0$ which is unique up to a constant factor. By normalizing, the stationary distribution π is obtained. The system of equations $xQ = 0$ is equivalent to the system of second-order vector-matrix difference equations

$$x_0 Q_{00} + x_1 Q_{10} = 0, \tag{2}$$

$$x_{n-1} Q_{n-1,n} + x_n Q_{nn} + x_{n+1} Q_{n+1,n} = 0, \qquad n = 1, \ldots, N-1, \tag{3}$$

$$x_{N-1} Q_{N-1,N} + x_N Q_{NN} = 0 \tag{4}$$

with row vectors $x_n \in \mathbb{R}^{d_n}$ and $x = (x_0, x_1, \ldots, x_n)$. If $Q_{n,n-1}^{-1}$ exists for all $n = 1, \ldots, N$, in particular $d_n = d$ for all $n = 0, \ldots, N$, then

$$x_1 = -x_0 Q_{00} Q_{10}^{-1}, \tag{5}$$

$$x_{n+1} = -x_{n-1} Q_{n-1,n} Q_{n+1,n}^{-1} - x_n Q_{nn} Q_{n+1,n}^{-1}, \quad n = 1, \ldots, N-1, \tag{6}$$

from which we can obtain $x_1 = x_0 F_1, \ldots, x_N = x_0 F_N$ with matrices $F_1, \ldots, F_N$. Substituting in the last equation yields $x_0(F_{N-1} Q_{N-1,N} + F_N Q_{NN}) = 0$ and we can compute x_0.

However, apart from the fact that $Q_{n,n-1}^{-1}$ does not exist in many applications, this algorithm of directly forward computing is numerically cumbersome and extremely unstable, because the solution of the difference equation $x_{n-1} Q_{n-1,n} + x_n Q_{nn} + x_{n+1} Q_{n+1,n} = 0$, which is compatible with the boundary conditions, is dominated by some other solutions. This holds as well for the similar backward computing algorithm. The MCF algorithm on the other hand transforms the second-order vector-matrix difference equation into a first-order recurrence scheme, that is $x_{n+1} = x_n R_n$ for $n = 0, \ldots, N-1$. Substituting this recursion into the difference equations yields

$$x_{N-1} \left(Q_{N-1,N} + R_{N-1} Q_{NN}\right) = 0, \tag{7}$$

$$x_{n-1} \left(Q_{n-1,n} + R_{n-1} Q_{nn} + R_{n-1} R_n Q_{n+1,n}\right) = 0, \qquad n = 1, \ldots, N-1, \tag{8}$$

$$x_0 \left(Q_{00} + R_0 Q_{10}\right) = 0. \tag{9}$$

The first equations can be used to compute recursively

$$R_{N-1} = -Q_{N-1,N}Q_{NN}^{-1}, \tag{10}$$

$$R_{n-1} = -Q_{n-1,n}\left(Q_{nn} + R_nQ_{n+1,n}\right)^{-1}, \quad n = 1,\ldots,N-1. \tag{11}$$

Then x_0 can be estimated by solving the last equation. By iterating $x_{n+1} = x_nR_n$ for $n = 0,\ldots,N-1$ we obtain the solution x and normalizing yields the stationary distribution.

The first similar algorithm for finite QBDs due to Gaver et. al. [4] is based on the backward recursion $x_n = x_{n+1}R_n$ leading to a forward recursion for the R_n. It has the disadvantage that it is not capable of solving infinite systems with, e.g., a state space truncation method or any other straightforward extension. Bright and Taylor [5] have developed an algorithm applicable to infinite level dependent QBDs and give a 'physical interpretation' of the matrices R_n by identifying the matrix coefficients as conditional expected state sojourn times. From that, it can be shown that Q_{NN}^{-1} and $\left(Q_{nn} + R_nQ_{n+1,n}\right)^{-1}$ exist as well as a nontrivial root x_0 of $x_0\left(Q_{00} + R_0Q_{10}\right) = 0$, which is unique up to a constant factor. Gaver et. al. [4] as well as Bright and Taylor [5] point out that for their methods there is a risk of overflow or underflow errors in the recursive computations of the matrices R_n such that numerical stability cannot be guaranteed.

An interpretation of the R_n as MCFs is due to Hanschke [10]. The main difference between a corresponding algorithm for the analysis of a QBD-type queueing model provided by Hanschke [10] and the method of Bright and Taylor [5] is the initial value for the recursive computation of the R_n if Q is infinite. While Bright and Taylor [5] attempt to approximate R_N before invoking the main algorithm, Hanschke [10] chooses $R_N = 0$ for a large N, which we adapt here for general level dependent QBDs. Altogether we get

Algorithm

- Compute $R_{N-1} = -Q_{N-1,N}Q_{NN}^{-1}$;
- Iterate $R_{n-1} = -Q_{n-1,n}\left(Q_{nn} + R_nQ_{n+1,n}\right)^{-1}$ for $n = N-1, N-2,\ldots,1$;
- Estimate a solution $\pi_0 \neq 0$ of $\pi_0\left(Q_{00} + R_0Q_{10}\right) = 0$;
- Iterate $\pi_{n+1} = \pi_nR_n$ for $n = 0,\ldots,N-1$;
- Renormalize π.

In practice, for the estimation of π_0 one column of $Q_{00} + R_0Q_{10}$ is simply ignored and one component of π_0 is fixed, for example $\pi_{01} = 1$. Note that the algorithm can be easily applied to QBDs with infinite state spaces. The algorithm can also be used if Q is the northwest-corner-truncation of an infinite generator matrix $\tilde{Q}$. An augmentation of the row sums to zero is not necessary if N is chosen large enough.

2.1. Memory-efficient implementation

For all examples we considered in a preliminary study the MCF algorithm is much faster than the Gaussian elimination and standard iterative methods (power method, Jacobi, Gauss-Seidel), but it turns out that with a too crude 'direct' implementation a disadvantage compared to the iterative methods is the larger memory requirement.

The matrix Q is often very sparse, that means in each row and in each column there are only m nonzero components with m about 3 to 6. This holds for the iteration matrix, too, and it preserve sparsity such that it can be stored using sparse matrix storage techniques, where the positions (4 bytes each for row index and column index) and the value of the matrix coefficient (8 bytes) is stored. In this manner each nonzero value requires 16 bytes. In our examples it is $d_0 = d_1 = d_2 = \ldots = d$, the number of blocks is $N+1$. Altogether the storage of the iteration matrix requires about $16 \cdot m \cdot (N+1) \cdot d$ bytes.

On the other hand for the MCF algorithm we have to store the matrices $R_0, R_1, \ldots, R_{N-1}$, which are usually not sparse and have to be stored in the standard way. Doing so and storing each value with 8 bytes we require $8 \cdot N \cdot d^2$ bytes for the MCF algorithm. Additionally, in both cases we require $8 \cdot (N+1) \cdot d$ bytes for the storage of the current approximation $\pi^{(n)}$ and the solution π respectively. Because this is the same for all algorithms we ignore this requirement in further comparison.

A simple idea for reducing the memory requirement of the MCF algorithm is not to store all the R_n but only $R_0, R_K, R_{2K}, \ldots$. R_0 is used to compute π_0 and π_1. Then starting with R_K we again compute $R_{K-1}, R_{K-2}, \ldots, R_1$ and $\pi_2, \ldots, \pi_{K+1}$ subsequently. Afterwards, starting with R_{2K} we compute $R_{2K-1}, R_{2K-2}, \ldots, R_{K+1}$ and $\pi_{K+2}, \ldots, \pi_{2K+1}$. In this way we only need memory for storing $K + \frac{N}{K}$ matrices at the same time. The number $K + \frac{N}{K}$ of course is small for $K \approx \sqrt{N}$ and this choice leads us to a memory requirement for $2\sqrt{N}$ matrices of size $d \times d$, that means about $16\sqrt{N}d^2$ bytes. If $m\sqrt{N} > d$ the MCF algorithm using this storage scheme requires less memory than the iterative methods. Of course, the method described above implies that the computing time grows. The worst case behavior would be a doubling of the computing time, but we will see in the examples that for $N >> d$ not that much additional time is required. Even with this storage scheme the MCF algorithm is much quicker than the iterative methods.

3. Numerical Examples

For the purpose of comparing the MCF algorithm with standard solution algorithms for Markov chains we have considered Gaussian elimination, the power method on the uniformized DTMC which has the same stationary distribution as the CTMC, the Jacobi iteration method, and the Gauss-Seidel iteration method, see, e.g., [6, 7] for details.

The methods were implemented in Matlab using the sparse matrix operations for all methods. The iterative methods were terminated as soon as the total difference $\left\|\pi^{(n+1)} - \pi^{(n)}\right\|$ of two successive iterates was smaller than 10^{-16}. We compared the computing time and the memory requirement dependent on the number of levels N and dependent on the block size $d = d_0 = d_1 = \ldots = d_N$.

3.1. Erlang/M/∞ Queue

Our first model is a specific PH/M/∞ queue, that is a queueing system with phase-type distributed interarrival times, exponential service times, and infinitely many servers. It can be modeled as a two-dimensional Markov chain with states (n, k) where n is the number of customers and k is the exponential phase of the interarrival time distribution for the next arriving customer. The phase-type distribution is specified by an initial distribution α and the generator matrix

$$\begin{pmatrix} T & t \\ 0 & 0 \end{pmatrix}$$

of an absorbing Markov chain. Using these notations we get $Q_{nn} = T - n\mu I$, $Q_{n,n+1} = t\alpha$ and $Q_{n,n-1} = n\mu I$. As a simple example we chose an Erlang distribution, defining $\alpha = (1, 0, \ldots, 0)$,

$$T = \begin{pmatrix} -\lambda & \lambda & & & \\ & -\lambda & \lambda & & \\ & & \ddots & \ddots & \\ & & & -\lambda & \lambda \\ & & & & -\lambda \end{pmatrix} \quad \text{and} \quad t = \begin{pmatrix} 0 \\ 0 \\ \vdots \\ 0 \\ \lambda \end{pmatrix}$$

with $\lambda = 8$ and $\mu = 3$. For the dependence on N we additionally fixed $d = 10$ and chose truncation levels N from 100 (Q has the size 1010×1010) to 1000 (Q has the size 10010×10010). For the dependence on d we fixed $N = 100$ and chose $d = 10, 20, 30, 40$.

Numerical results for different truncation levels and different numbers of phases are depicted in Figures 1–4. In particular, we note that the Jacobi iteration method did not converge at all. Terminating the iteration after 60 seconds for $N = 100$ produced an error of $\|\pi Q\| \approx 0.03$. Even direct forward computing produces a smaller error of about 10^{-7}. Direct backward computing is not possible for $Q^{-1}_{n,n+1}$ does not exist.

3.2. A Telecommunications model with impatient customers

Our next model is a telecommunications model with impatient customers that was also discussed, e.g., in [6]. It deals with the influence of impatience on telephone customers during a computerized telephone exchange. Customers arrive at the telecommunication server forming a Poisson process with rate a. At the server, customers are processed according to the processor sharing discipline with maximum capacity N. Customers arriving at a full server are lost. If there are $n > 0$ customers at the server, one of them may leave the server either after being successfully served (rate

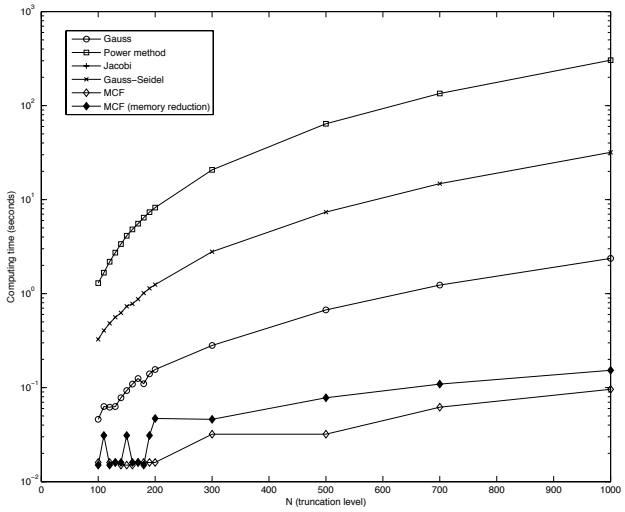

Figure 1: Computing times versus truncation level for the Erlang/M/∞ model

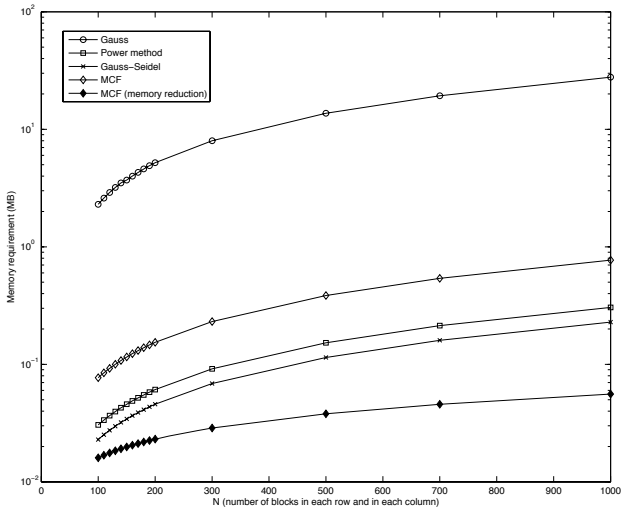

Figure 2: Memory requirements versus truncation level for the Erlang/M/∞ model

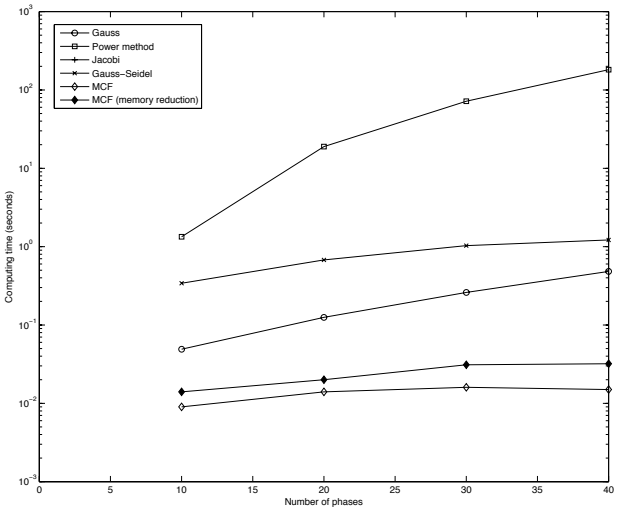

Figure 3: Computing times versus number of phases for the Erlang/M/∞ model

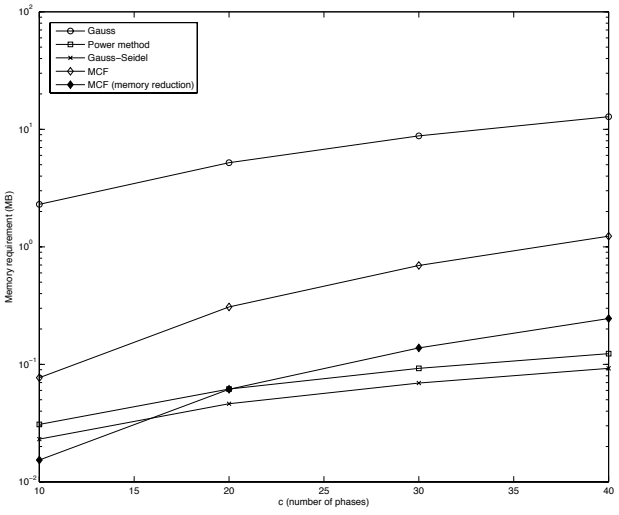

Figure 4: Memory requirements versus number of phases for the Erlang/M/∞ model

μ) or due to impatience (rate $n \cdot \tau$). An impatient customer may quit completely (with a probability $1 - h$), or, after an exponentially distributed 'thinking time' (rate λ), rejoin the telecommunication server.

If the thinking time is interpreted as a $\bullet/M/\infty$-node, the system is transformed into a two-node open queuing network. The states of the system are described by pairs $(n, k) \in \{0, \ldots, N\} \times \mathbb{N}_0$, where n is the number of customers being served and k is the number of customers 'thinking'. In order to obtain a finite Markov chain we additionally assume $k \leq s$, where s has to be chosen large enough. The rates of the corresponding Markov chain are as follows.

- new arrival: $q_{(n,k),(n+1,k)} = a$, $n \leq N$,
- successfully served or leaving the network due to impatience: $q_{(n,k),(n-1,k)} = \mu + n\tau(1-h)$, $n \geq 1$,
- leaving the server due to impatience and 'begin thinking': $q_{(n,k),(n-1,k+1)} = n\tau h$, $n \geq 1$, $k \leq s-1$,
- rejoining the server after 'thinking': $q_{(n,k),(n+1,k-1)} = k\lambda$, $n \leq N$, $k \geq 1$.

The diagonal entries of Q, $Q_{(n,k),(n,k)}$ guarantee that the row sums of Q disappear, all other neglected values are zero. The desired structure of Q is achieved by

$$Q_{n,n-1} = (q_{(n,k),(n-1,\ell)})_{k,\ell=1}^{s}, \tag{12}$$

$$Q_{n,n} = (q_{(n,k),(n,\ell)})_{k,\ell=1}^{s}, \tag{13}$$

$$Q_{n,n+1} = (q_{(n,k),(n+1,\ell)})_{k,\ell=1}^{s}. \tag{14}$$

We chose $a = 0.6$, $\mu = 1$, $\tau = 0.05$, $h = 0.85$ and $\lambda = 5$ and $s = 10$ (dependence on N) respectively $N = 550$ (dependence on s). Numerical results for different numbers of blocks and different block sizes are depicted in

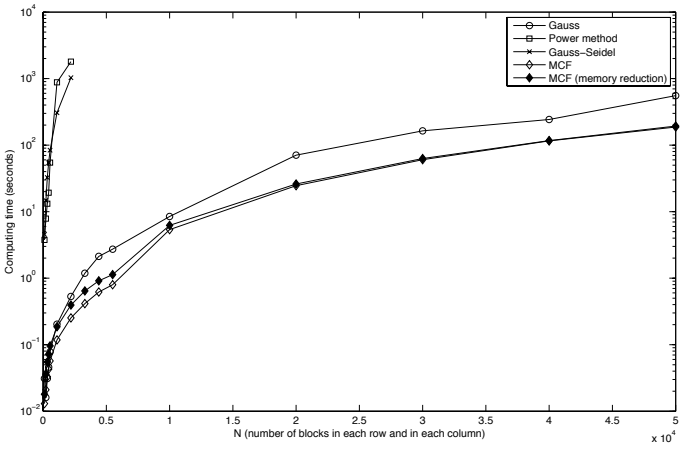

Figure 5: Computing times versus number of blocks for the telecommunications model

Figures 5–8. The iterative methods converged very slowly or did not converge at all, e.g., for $s = 10$ and $N = 110, 220, 330, 440, 550$ we terminated the Jacobi method after 180 seconds and the error was still large (between 10^{-3} and 10^{-2}). The other iterative methods converged for small values of N in reasonable time and the errors $\|\pi Q\|$ were about 10^{-13} for the power method and smaller than 10^{-15} for the Gauss-Seidel method, the Gaussian elimination and the MCF algorithm. For larger N we only compared the MCF with the Gaussian elimination. For the largest model in this example ($N = 50000$ and $s = 10$) Q is a 550011×550011-matrix. The direct solution methods for matrix-vector difference equations in this example for $N = 110$ and $s = 10$ produced errors $\|\pi Q\|$ of about 51 (forward computing) and 1.3 (backward computing).

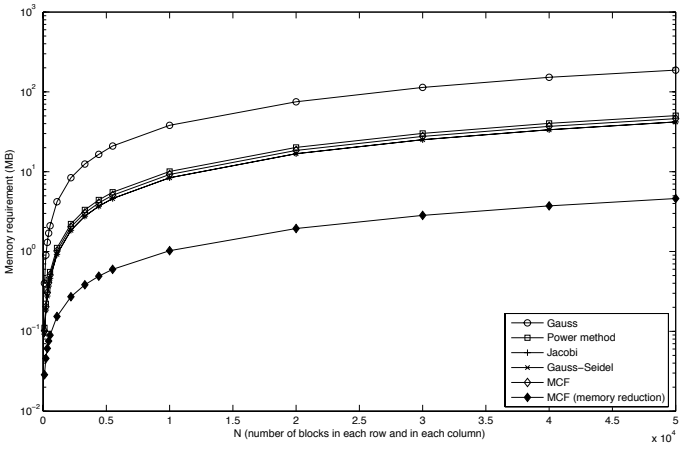

Figure 6: Memory requirements versus number of blocks for the telecommunications model

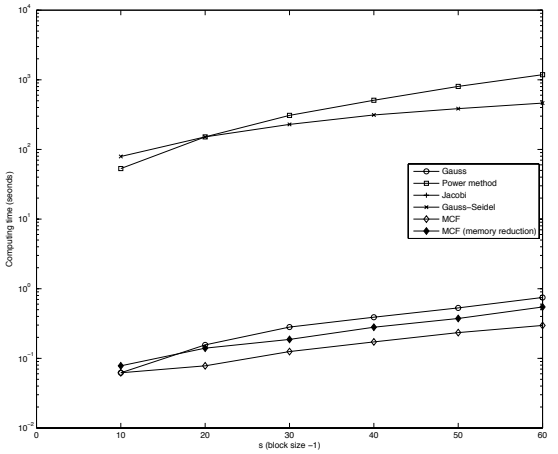

Figure 7: Computing times versus block size for the telecommunications model

4. Conclusion

We have addressed the numerical computation of stationary distributions for level dependent QBDs. A matrix continued fraction (MCF) algorithm has been proposed and compared with standard solution techniques for Markov chains. It turns out that the MCF algorithm is much faster than the standard approaches and numerically stable.

Further research includes more extensive empirical comparisons with other algorithms such as aggregation tech-

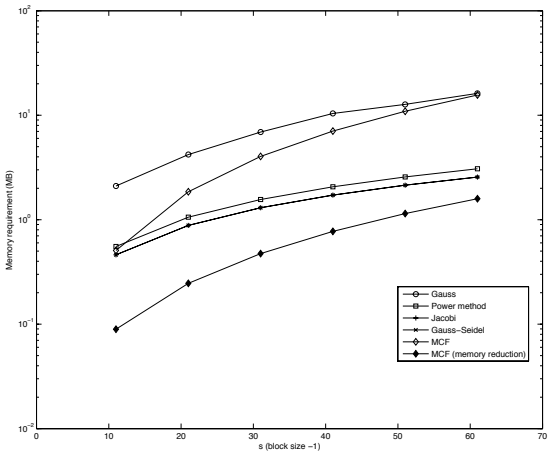

Figure 8: Memory requirements versus block size for the telecommunications model

niques and the Bright-Taylor (BT) algorithm. Also the major similarities and differences between the MCF algorithm and the BT algorithm should be worked out. Furthermore, more complex models including state spaces of dimension higher than two are under consideration. This fits the framework as we only require the desired block-tridiagonal structure of the generator or transition probability matrix and do not require restrictions on the dimensionality. In particular, the application to queueing networks and epidemiological models is currently under investigation.

Another topic of further research is the extension of the algorithm beyond QBDs in order to be able to efficiently analyze Markov chains with more general transition structure such as QBDs with catastrophes or Markov chains skip-free to the right having block lower Hessenberg generator or transition probability matrices.

References

[1] M. F. Neuts, Matrix-Geometric Solutions in Stochastic Models, The John Hopkins University Press, 1981.
[2] G. Latouche, V. Ramaswami, A logarithmic reduction algorithm for quasi birth and death processes, Journal of Applied Probability 30 (1993) 650–674.
[3] G. Latouche, V. Ramaswami, Introduction to Matrix Analytic Methods in Stochastic Modeling, SIAM, 1999.
[4] D. P. Gaver, P. A. Jacobs, G. Latouche, Finite birth-and-death models in randomly changing environments, Advances in Applied Probability 16 (4) (1984) 715–731.
[5] L. Bright, P. G. Taylor, Calculating the equilibrium distribution in level dependent quasi-birth-and-death processes, Stochastic Models 11 (3) (1995) 497–525.
[6] B. Philippe, Y. Saad, W. J. Stewart, Numerical methods in Markov chain modelling, Operations Research 40 (6) (1992) 1156–1179.
[7] W. J. Stewart, Introduction to the Numerical Solution of Markov Chains, Princeton University Press, 1994.
[8] P. Levrie, A. Bultheel, Matrix continued fractions related to first-order linear recurrence systems, Electronical Transactions on Numerical Analysis 4 (1996) 46–63.
[9] T. Hanschke, Markov chains and generalized continued fractions, Journal of Applied Probability 29 (4) (1992) 838–849.
[10] T. Hanschke, A matrix continued fraction algorithm for the multiserver repeated order queue, Mathematical and Computer Modelling 30 (1999) 159–170.
[11] M. Raissouli, A. Kacha, Convergence of matrix continued fractions, Linear Algebra and its Applications 320 (1–3) (2000) 115–129.
[12] V. N. Sorokin, J. Van Iseghem, Matrix continued fractions, Journal of Approximation Theory 96 (2) (1999) 237–257.
[13] H. Zhao, G. Zhu, Matrix-valued continued fractions, Journal of Approximation Theory 120 (1) (2003) 136–152.