Improved Cyclic Reduction for Solving Queueing Problems

Dario Andrea Bini and Beatrice Meini
Dipartimento di Matematica
Università di Pisa
Via Buonarroti 2, 56127 Pisa, Italy
bini@dm.unipi.it, meini@dm.unipi.it

Abstract

The cyclic reduction technique [7], rephrased in functional form [3], provides a numerically stable, quadratically convergent method, for solving the matrix equation $X = \sum_{i=0}^{+\infty} X^i A_i$, where the A_i's are nonnegative $k \times k$ matrices such that $\sum_{i=0}^{+\infty} A_i$ is column stochastic. In this paper we propose a further improvement of the above method, based on a pointwise evaluation/interpolation at a suitable set of Fourier points, of the functional relations defining each step of cyclic reduction [3]. This new technique allows us to devise an algorithm based on FFT having a lower computational cost and a higher numerical stability. Numerical results and comparisons are provided.

Keywords. Queueing problems, M/G/1 type matrices, cyclic reduction, Toeplitz matrices, FFT.

1 Introduction

The mathematical modelling of many queueing problems leads to solving the infinite system

$$(I - P)\boldsymbol{\pi} = 0, ||\boldsymbol{\pi}||_1 = 1, \tag{1}$$

where P is a column stochastic matrix in block Hessenberg form having the block Toeplitz-like structure

$$P = \begin{pmatrix} B_1 & A_0 & & \bigcirc & \\ B_2 & A_1 & A_0 & & \\ B_3 & A_2 & A_1 & A_0 & \\ \vdots & \vdots & \ddots & \ddots & \ddots \end{pmatrix} \tag{2}$$

and the blocks A_i, B_i are nonnegative $k \times k$ matrices such that $\sum_{i=0}^{\infty} A_i$, $\sum_{i=1}^{\infty} B_i$ are column stochastic. Matrices of the structure (2) are known in literature as

stochastic matrices of M/G/1 type [16]. It is well known that the solution of (1) exists uniquely if the matrix P is irreducible and positive recurrent [8], moreover the computation of $\boldsymbol{\pi}$ can be obtained from the computation of the minimal nonnegative solution G of the nonlinear matrix equation

$$X = \sum_{i=0}^{\infty} X^i A_i, \tag{3}$$

by means of the Ramaswami formula [16], [17], [14], where X is a $k \times k$ matrix. This makes of great interest the study of efficient numerical methods for solving equation (3).

Recently, the analysis of numerical methods for the computation of the solution G of (3) has been largely developed. Classical linearly convergent methods, based on functional iteration techniques, have been studied in detail in [15], [10], [16] and [18]. New quadratically convergent methods have been proposed in [3], [13], [11], [20]. These methods show their effectiveness specially in the cases where classical algorithms based on functional iterations converge very slowly [3]. From the numerical experiments performed so far [1], [3], the most efficient method seems to be the one of [3]. Indeed, the methods proposed in [13], [11], even if quadratically convergent, have a much higher computational cost per iteration.

The method of [3] consists in rewriting the matrix equation (3) in matrix form as

$$(X, X^2, X^3, \ldots) \begin{pmatrix} I - A_1 & -A_0 & & \bigcirc & \\ -A_2 & I - A_1 & -A_0 & & \\ -A_3 & -A_2 & I - A_1 & -A_0 & \\ \vdots & \vdots & \ddots & \ddots & \ddots \end{pmatrix} = (A_0, O, O, \ldots) \tag{4}$$

and in applying the cyclic reduction technique, rephrased in functional form, to solve (4). A single step of cyclic reduction consists in performing an odd-even permutation of the block rows and the block columns of the matrix (4) followed by one step of block Gaussian elimination. It is shown in [3] that this transformation keeps unchanged the structure of the system. By means of a recursive application of cyclic reduction, it is generated a sequence of block Hessenberg, block Toeplitz-like infinite systems that quadratically converges to a limit system whose solution can be explicitely evaluated.

The matrix of the system at step n, having the same block Toeplitz-like, block Hessenberg structure, of (2), is fully defined by its first and second block columns. The block entries of the first block column, denoted by $I - \widehat{A}_1^{(n)}$, $-\widehat{A}_2^{(n)}$, $-\widehat{A}_3^{(n)}, \ldots$, and the block entries of the second block column, denoted by $-A_0^{(n)}$, $I - A_1^{(n)}$, $-A_2^{(n)}, \ldots$, define two formal matrix power series $\widehat{\varphi}^{(n)}(z) = \sum_{i=0}^{+\infty} \widehat{A}_{i+1}^{(n)} z^i$, $\varphi^{(n)}(z) = \sum_{i=0}^{+\infty} A_i^{(n)} z^i$ associated with the system. Moreover, an

explicit functional relation intercurs between $\varphi^{(n+1)}(z)$, $\widehat{\varphi}^{(n+1)}(z)$ and $\varphi^{(n)}(z)$, $\widehat{\varphi}^{(n)}(z)$:

$$(\varphi^{(n+1)}(z), \widehat{\varphi}^{(n+1)}(z)) = F(\varphi^{(n)}(z), \widehat{\varphi}^{(n)}(z)) \tag{5}$$

for a suitable F, which expresses in functional form the cyclic reduction step.

The functional representation (5) allows one to reduce the overall computation for performing formal operations between matrix power series, which are numerically truncated to matrix polynomials, i.e., polynomials having matrix coefficients. The use of fast polynomial arithmetic, based on FFT [4], [5], for computing products and reciprocals of polynomials, provides a tool for dramatically reducing the computational cost, arriving at an efficient and reliable implementation of the algorithm [3].

In this paper we propose an improvement of the algorithm of [3], based on a more efficient implementation of the functional relations (5) which the algorithm relies on. More precisely, the idea at the basis of our improvement consists in a pointwise evaluation of (5) at a suitable set of Fourier points. From these values we may recover the block coefficients of the matrix series $\varphi^{(n+1)}(z)$, $\widehat{\varphi}^{(n+1)}(z)$ once we are given the block coefficients of $\varphi^{(n)}(z)$, $\widehat{\varphi}^{(n)}(z)$. This evaluation/interpolation procedure provides accurate approximations of the matrix power series if the number of Fourier points is large enough. In fact, we introduce a modification of the FFT algorithm which allows us to compute the values of $\varphi^{(n)}(z)$, $\widehat{\varphi}^{(n)}(z)$ at the even-indexed Fourier points and, if needed, to compute the values of these power series taken on at the odd-indexed Fourier points without having to recompute all the values from scratch.

In order to determine the number of Fourier points sufficient to reach the desired accuracy, we introduce a criterium allowing us to control the approximation error and to adjust dynamically the number of Fourier points at each step of the algorithm with no waste of computation.

By using this different strategy we arrive at a new version of the algorithm which, besides being much easier to implement, leads to a reduction of the computational cost of a factor of about 2 and is numerically more stable.

The paper is organized as follows. In Section 2 we recall the algorithm based on cyclic reduction and the main convergence results. In Section 3 we propose the new algorithm based on the pointwise evaluation of (5). In Section 4 we introduce the basic tools for FFT computations. In Section 5 we describe the implementation of the algorithm in Fortran 90 and present numerical results.

2 The cyclic reduction algorithm.

In this section we recall the main results concerning the cyclic reduction method applied to solve equation (3). For major details we refer the reader to [3] and [2].

The key idea consists in rewriting the nonlinear equation (3) in the matrix form (4). In order to solve the block Hessenberg, block Toeplitz infinite system

(4) we recursively apply the cyclic reduction algorithm [7], i.e. a block odd-even permutation followed by one step of Gaussian elimination, thus generating the sequence of block Hessenberg, block Toeplitz-like infinite systems

$$(X, X^{2^n+1}, X^{2\cdot 2^n+1}, \ldots) \begin{pmatrix} I-\widehat{A}_1^{(n)} & -A_0^{(n)} & & \bigcirc & \\ -\widehat{A}_2^{(n)} & I-A_1^{(n)} & -A_0^{(n)} & & \\ -\widehat{A}_3^{(n)} & -A_2^{(n)} & I-A_1^{(n)} & -A_0^{(n)} & \\ \vdots & \vdots & \ddots & \ddots & \ddots \end{pmatrix} =$$

$$(A_0, O, O, \ldots),\ n \geq 0,$$

$$A_0^{(0)} = A_0,\ \widehat{A}_i^{(0)} = A_i^{(0)} = A_i,\ i \geq 1. \tag{6}$$

The block entries $\{A_i^{(n+1)}\}_{i\geq 0}$, $\{\widehat{A}_i^{(n+1)}\}_{i\geq 1}$, obtained at step $n+1$, are related to the block entries $\{A_i^{(n)}\}_{i\geq 0}$, $\{\widehat{A}_i^{(n)}\}_{i\geq 1}$, obtained at step n, by means of functional relations, that express in compact form the block odd-even permutation and the Gaussian elimination of one step of cyclic reduction. More precisely, for any $n \geq 0$, let us associate with the matrix sequences $\{A_i^{(n)}\}_{i\geq 0}$, $\{\widehat{A}_i^{(n)}\}_{i\geq 1}$, the formal matrix power series $\varphi^{(n)}(z) = \sum_{i=0}^{+\infty} A_i^{(n)} z^i$, $\widehat{\varphi}^{(n)}(z) = \sum_{i=0}^{+\infty} \widehat{A}_{i+1}^{(n)} z^i$, respectively. Then the following functional relations hold:

$$\begin{cases} \varphi^{(n+1)}(z) = z\varphi_{odd}^{(n)}(z) + \varphi_{even}^{(n)}(z)\left(I - \varphi_{odd}^{(n)}(z)\right)^{-1} \varphi_{even}^{(n)}(z) \\ \widehat{\varphi}^{(n+1)}(z) = \widehat{\varphi}_{odd}^{(n)}(z) + \varphi_{even}^{(n)}(z)(I - \varphi_{odd}^{(n)}(z))^{-1} \widehat{\varphi}_{even}^{(n)}(z) \end{cases} \tag{7}$$

where

$$\begin{array}{ll} \varphi_{even}^{(n)}(z) = \sum_{i=0}^{+\infty} A_{2i}^{(n)} z^i, & \varphi_{odd}^{(n)}(z) = \sum_{i=0}^{+\infty} A_{2i+1}^{(n)} z^i \\ \widehat{\varphi}_{even}^{(n)}(z) = \sum_{i=0}^{+\infty} \widehat{A}_{2(i+1)}^{(n)} z^i, & \widehat{\varphi}_{odd}^{(n)}(z) = \sum_{i=0}^{+\infty} \widehat{A}_{2i+1}^{(n)} z^i. \end{array} \tag{8}$$

The relations (7), besides to express in compact form the recursions which the cyclic reduction method is based on, provide a basic tool for the efficient computation of the matrices $\{A_i^{(n)}\}_{i\geq 0}$, $\{\widehat{A}_i^{(n)}\}_{i\geq 1}$, as we will explain in the next section.

The following results are fundamental for devising an efficient algorithm based on cyclic reduction:

Theorem 1 *The blocks $\{A_i^{(n)}\}_{i\geq 0}$, $\{\widehat{A}_i^{(n)}\}_{i\geq 1}$ are nonnegative matrices such that $\sum_{i=0}^{+\infty} A_i^{(n)}$, $A_0 + \sum_{i=1}^{+\infty} \widehat{A}_i^{(n)}$ are stochastic. Moreover, if the matrix $I - \sum_{i=0}^{+\infty} G^{i\cdot 2^j} \widehat{A}_{i+1}^{(n)}$ is nonsingular, then*

$$G = A_0 \left(I - \sum_{i=0}^{+\infty} G^{i\cdot 2^j} \widehat{A}_{i+1}^{(n)} \right)^{-1}. \tag{9}$$

Under mild conditions, usually satisfied in the applications, the block entries $A_i^{(n)}$ and $\widehat{A}_i^{(n)}$ quadratically converge to zero, for $i \geq 2$, for $n \to \infty$, as stated by the following theorem [3]:

Theorem 2 *Let $G' = \lim_{n\to\infty} G^n$. Then the following convergence properties hold:*

1. $\lim_{n\to\infty} \widehat{A}_i^{(n)} = 0$, *for* $i \geq 2$;
2. *if the entries of the matrix* $(I - \sum_{i=1}^{+\infty} A_i^{(n)})^{-1}$ *are bounded above by a constant, then the sequence of matrices*

$$R^{(n)} = A_0^{(n)} \left(I - \sum_{i=1}^{+\infty} A_i^{(n)} \right)^{-1} \tag{10}$$

   converges quadratically to the matrix G';
3. *if the solution G of (3) is irreducible then*

$$\lim_{n\to\infty} A_0^{(n)} (I - A_1^{(n)})^{-1} = G'$$
$$\lim_{n\to\infty} A_i^{(n)} = 0, \ i \geq 2.$$

Under the assumptions of theorems 1 and 2 the following algorithm for the numerical computation of the matrix G can be applied.

Algorithm 2.1

1. Apply cyclic reduction to (4), by means of (7), obtaining the sequence (6) of infinite systems defined by the blocks $A_i^{(n)}$,$\widehat{A}_i^{(n)}$, $n = 1, 2, \ldots, q$, until one of the following conditions is satisfied

   (C1) $|R^{(q)} - R^{(q-1)}| < \epsilon E$,

   (C2) $\mathbf{e}^T (I - A_0^{(q)} (I - A_1^{(q)})^{-1}) < \epsilon \mathbf{e}^T$,

   (C3) $\mathbf{e}^T (I - A_0 (I - \widehat{A}_1^{(q)})^{-1}) < \epsilon \mathbf{e}^T$,

   where, at each step n, the matrix $R^{(n)}$ is defined in (10), $\epsilon > 0$ is fixed and E is the $k \times k$ matrix having all the entries equal to 1.

2. Compute an approximation of the matrix G:

   (a) if condition **(C1)** or condition **(C2)** is verified compute an approximation of G by replacing, in the right hand side of (9), for $n = q$, the positive powers of G with $R^{(q)}$;

   (b) if condition **(C3)** is verified an approximation of G is given by $A_0 (I - \widehat{A}_1^{(q)})^{-1}$.

3 Pointwise Cyclic Reduction

The effectiveness of algorithm 2.1 relies on the possibility of computing the blocks $A_i^{(n)}$, $\widehat{A}_i^{(n)}$, at each step n in an efficient way: an improved computation of such matrices can lead to a strong reduction of the computational cost of each iteration.

According to the ideas developed in [3] and [2], we may truncate the sequences $\{A_i^{(n)}\}_i$ $\{\widehat{A}_i^{(n)}\}_i$ to the index m_n and $\widehat{m}_n$, respectively, such that $\mathbf{e}^T(I - \sum_{i=0}^{m_n} A_i^{(n)}) < \sigma\mathbf{e}^T$ and $\mathbf{e}^T(I - A_0 - \sum_{i=1}^{\widehat{m}_n} \widehat{A}_i^{(n)}) < \sigma\mathbf{e}^T$, where σ is an upper bound to the machine precision of the floating point arithmetic used in the computation, and is related to the accuracy of the computed approximation of G. In other words, due to theorem 1, we consider only those blocks which make the matrices $\sum_{i=0}^{m_n} A_i^{(n)}$ and $A_0 + \sum_{i=1}^{\widehat{m}_n} \widehat{A}_i^{(n)}$ numerically stochastic with respect to σ. A nonnegative matrix A is said *numerically stochastic with respect to* σ, or σ-*stochastic*, if $\mathbf{e}^T(I - A) < \sigma\mathbf{e}^T$. In this way, the matrix series $\varphi^{(n)}(z)$, $\widehat{\varphi}^{(n)}(z)$ are replaced by matrix polynomials of degree m_n, $\widehat{m}_n - 1$, respectively, that we denote with the same symbols $\varphi^{(n)}(z)$, $\widehat{\varphi}^{(n)}(z)$. The degrees m_n and $\widehat{m}_n - 1$ are called the *numerical degrees* of the series $\varphi^{(n)}(z)$, $\widehat{\varphi}^{(n)}(z)$, respectively. Thus, relations (7) can be rewritten in terms of matrix polynomials, in the variable z, instead of matrix power series, and all the computations can be performed by means of techniques based on Fast Fourier Transform, drastically reducing the computational cost.

In [2] and [3] we have extended to matrix polynomials the FFT-based techniques which have been devised for scalar polynomial arithmetic [5], [4], and we have reduced all the operations involved in (7) to multiply matrix polynomials and invert a matrix polynomial modulo z^m, for a suitable m (a detailed description of the algorithms is provided in [3]). In this way the computational cost to perform one step of cyclic reduction is reduced to $O(k^3 M_n + k^2 M_n \log M_n)$ arithmetic operations, where M_n is such that the matrices $\sum_{i=0}^{M_n} A_i^{(n)}$ and $A_0 + \sum_{i=1}^{M_n} \widehat{A}_i^{(n)}$ are numerically stochastic, versus the cost of $O(k^3 M_n^2)$ if the customary arithmetic were used.

In this section we propose a new improved method for the computation of one step of cyclic reduction, based on a point-wise evaluation at the roots of 1, of the series involved in (7). The new approach consists in performing the following steps:

1. Evaluating the series $\varphi_{odd}^{(n)}(z)$, $\varphi_{even}^{(n)}(z)$, $\widehat{\varphi}_{odd}^{(n)}(z)$, $\widehat{\varphi}_{even}^{(n)}(z)$ at the M_n-th roots of 1, where $M_n - 1 = 2^p - 1$ is an upper bound to the numerical degree of the above series.

2. Performing a pointwise evaluation of (7) at the M_n-th roots of 1.

3. Computing the coefficients of the matrix polynomials $P(z)$ and $\widehat{P}(z)$ of

degree $M_n - 1$ which interpolate the values of the matrix series $\varphi^{(n+1)}(z)$, $\widehat{\varphi}^{(n+1)}(z)$ obtained at stage 2.

4. Checking whether or not the matrix polynomials $P(z)$ and $\widehat{P}(z)$ are good approximations of the series $\varphi^{(n+1)}(z)$, $\widehat{\varphi}^{(n+1)}(z)$, respectively.

5. If the polynomials $P(z)$ and $\widehat{P}(z)$ are poor approximations of $\varphi^{(n+1)}(z)$, $\widehat{\varphi}^{(n+1)}(z)$, set $M_n = 2M_n$ and repeat steps 1–5 until the accuracy of the approximation is reached.

Due to the properties of FFT, in each doubling step, part of the results is already available at no cost. Moreover, unlike in the implementation of the algorithm of [2] and [3], the computation of the reciprocal of a polynomial is avoided and the order of the involved DFT and IDFT is kept to its minimum value, thus substantially reducing the computational cost per iteration.

The following properties of cyclic reduction and of FFT provide a good test to check the accuracy of the approximations $P(z)$, $\widehat{P}(z)$ of the series $\varphi^{(n+1)}(z)$, $\widehat{\varphi}^{(n+1)}(z)$ needed at step 4:

Theorem 3 *Let, for any $n \geq 0$,*

$$\boldsymbol{\alpha}^{(n)T} = \mathbf{e}^T \sum_{i=1}^{\infty} iA_i^{(n)}, \;\; \widehat{\boldsymbol{\alpha}}^{(n)T} = \mathbf{e}^T \sum_{i=1}^{\infty} i\widehat{A}_{i+1}^{(n)}.$$

Then the following recursive relations hold:

$$\begin{aligned} \boldsymbol{\alpha}^{(n+1)T} &= \left(\mathbf{e}^T + \boldsymbol{\alpha}^{(n)T} - (\mathbf{e}^T - \boldsymbol{\alpha}^{(n)T})(I - \varphi_{odd}^{(n)}(1))^{-1}\varphi_{even}^{(n)}(1)\right)/2 \\ \widehat{\boldsymbol{\alpha}}^{(n+1)T} &= \left(\widehat{\boldsymbol{\alpha}}^{(n)T} - (\mathbf{e}^T - \boldsymbol{\alpha}^{(n)T})(I - \varphi_{odd}^{(n)}(1))^{-1}\widehat{\varphi}_{even}^{(n)}(1)\right)/2. \end{aligned} \tag{11}$$

Proof. Observe that, at each step n, the following relation holds:

$$(\mathbf{0}, \mathbf{e}^T, 2\mathbf{e}^T, 3\mathbf{e}^T, \ldots) \begin{pmatrix} I - \widehat{A}_1^{(n)} & -A_0^{(n)} & & \bigcirc & \\ -\widehat{A}_2^{(n)} & I - A_1^{(n)} & -A_0^{(n)} & & \\ -\widehat{A}_3^{(n)} & -A_2^{(n)} & I - A_1^{(n)} & -A_0^{(n)} & \\ \vdots & \vdots & \ddots & \ddots & \ddots \end{pmatrix} =$$

$$(-\widehat{\boldsymbol{\alpha}}^{(n)T}, \mathbf{e}^T - \boldsymbol{\alpha}^{(n)T}, \mathbf{e}^T - \boldsymbol{\alpha}^{(n)T}, \mathbf{e}^T - \boldsymbol{\alpha}^{(n)T}, \ldots).$$

By applying to the above system one step of cyclic reduction we easily obtain formulae (11).

Proposition 4 *Let $s(z) = \sum_{i=0}^{\infty} s_i z^i$ be a series converging in the closed unit disk. Let $r^{(m)}(z) = \sum_{i=0}^{m-1} r_i z^i$ be the polynomial of degree $m-1$ interpolating*

the series $s(z)$ at the m-th roots of 1. Then, for $i = 0, \ldots, m-1$, the coefficients r_i are given by

$$r_i = \sum_{j=0}^{\infty} s_{i+jm}.$$

Proof Let ω be a principal m-th root of 1. Then the coefficient r_i, for $i = 0, \ldots, m-1$, is given by

$$r_i = \frac{1}{m} \sum_{h=0}^{m-1} \omega^{-ih} s(\omega^h) = \frac{1}{m} \sum_{h=0}^{m-1} \sum_{j=0}^{\infty} \omega^{(-i+j)h} s_j = \sum_{j=0}^{\infty} s_{i+jm}.$$

The above proposition can be easily extended to the case of matrix power series, thus leading to the following

Theorem 5 *Let, at step n, $P^{(m)}(z) = \sum_{i=0}^{m-1} P_i z^i$ and $\widehat{P}^{(m)} = \sum_{i=0}^{m-1} \widehat{P}_i z^i$ be the matrix polynomials of degree $m-1$ interpolating the series $\varphi^{(n)}(z)$, $\widehat{\varphi}^{(n)}(z)$ at the m-th roots of 1. Then the following inequalities hold:*

$$\mathbf{e}^T \textstyle\sum_{i=m}^{\infty} A_i^{(n)} \leq \boldsymbol{\alpha}^{(n)T} - \mathbf{e}^T \sum_{i=1}^{m-1} iP_i$$

$$\mathbf{e}^T \textstyle\sum_{i=m+1}^{\infty} \widehat{A}_i^{(n)} \leq \widehat{\boldsymbol{\alpha}}^{(n)T} - \mathbf{e}^T \sum_{i=1}^{m-1} i\widehat{P}_i.$$

Proof The proof readily follows from Proposition 4 and from the definition of $\boldsymbol{\alpha}^{(n)}$ and $\widehat{\boldsymbol{\alpha}}^{(n)}$ given in theorem 3.

Theorems 3 and 5 provide a good test to check the accuracy of the approximations of the series $\varphi^{(n)}(z)$, $\widehat{\varphi}^{(n)}(z)$ at each step n. Indeed, suppose to know the coefficients of the series $\varphi^{(n)}(z)$, $\widehat{\varphi}^{(n)}(z)$, and to compute the coefficients P_i, $\widehat{P}_i$, $i = 0, \ldots, m-1$, of the approximations $P^{(m)}(z) = \sum_{i=0}^{m-1} P_i z^i$, $\widehat{P}^{(m)}(z) = \sum_{i=0}^{m-1} \widehat{P}_i z^i$ of the series $\varphi^{(n+1)}(z)$, $\widehat{\varphi}^{(n+1)}(z)$ by interpolating the functional relations (7) at the m-th roots of 1. From Theorem 5 it follows that, if

$$\boldsymbol{\alpha}^{(n+1)T} - \mathbf{e}^T \sum_{i=1}^{m-1} iP_i \leq \epsilon \mathbf{e}^T \tag{12}$$

and

$$\widehat{\boldsymbol{\alpha}}^{(n+1)T} - \mathbf{e}^T \sum_{i=1}^{m-1} i\widehat{P}_i \leq \epsilon \mathbf{e}^T, \tag{13}$$

then the matrices $\sum_{i=0}^{m-1} A_i^{(n+1)}$ and $A_0 + \sum_{i=1}^{m} \widehat{A}_i^{(n+1)}$ are ϵ–stochastic. Whence, for ϵ "small enough", the series $\varphi^{(n+1)}(z)$, $\widehat{\varphi}^{(n+1)}(z)$, can be truncated to polynomials of degree $m-1$, and $P^{(m)}(z)$, $\widehat{P}^{(m)}(z)$ are good approximations. It is important to point out that (12) and (13) can be easily applied without knowing the coefficients of the series $\varphi^{(n+1)}(z)$, $\widehat{\varphi}^{(n+1)}(z)$. In fact, $\boldsymbol{\alpha}^{(n+1)}$ and $\widehat{\boldsymbol{\alpha}}^{(n+1)}$ can be explicitely obtained by means of (11) at the cost of $O(k^3)$ arithmetic operations. This provides an efficient tool to check the accuracy of the approximations $P^{(m)}(z)$, $\widehat{P}^{(m)}(z)$ to $\varphi^{(n+1)}(z)$, $\widehat{\varphi}^{(n+1)}(z)$.

4 Computing DFT's.

The efficient computation of the functions (7) relies on the possibility of evaluating/interpolating a matrix power series, which is numerically reduced to a matrix polynomial, at the roots of 1 by means of FFT. In this section we introduce and analyze some tools and computations needed for dealing with this problem. Firstly, we recall well-known formulae for the computation of DFT's and IDFT's of real vectors [6], [9]. Then we provide suitable formulae for similar computations where part of the result is already available at no cost.

Throughout this section $\mathbf{i}$ is the imaginary unit such that $\mathbf{i}^2 = -1$, $\omega_m = \cos\frac{2\pi}{m} + \mathbf{i}\sin\frac{2\pi}{m}$ denotes a principal m-th root of 1, $\bar{x}$ denotes the complex conjugate of $x \in \mathbf{C}$. The vector $\mathbf{b} = (b_j) \in \mathbf{C}^m$ such that $b_j = \sum_{i=0}^{m-1} a_i \omega_m^{ij}$ is called the discrete Fourier transform of the vector $\mathbf{a} = (a_j)$, and is denoted by $\mathbf{b} = \mathrm{DFT}_m(\mathbf{a})$, the vector $\mathbf{a} = \mathrm{IDFT}_m(\mathbf{b})$, $\mathbf{a} = (a_j)$, $a_j = \frac{1}{m}\sum_{i=0}^{m-1} \bar{\omega}_m^{ij} b_i$ denotes the inverse discrete Fourier transform of $\mathbf{b}$.

We recall two known equations relating the vectors $\mathbf{w}, \mathbf{z} \in \mathbf{C}^m$ such that $\mathbf{z} = \mathrm{DFT}_m(\mathbf{w})$, on which the algorithms of Cooley-Tukey and Sande-Tukey are based. Let

$$\begin{array}{l} D = \mathrm{diag}(1, \omega_m, \omega_m^2, \ldots, \omega_m^{m/2-1}), \\ \mathbf{w}^{(1)} = (w_j),\ \mathbf{w}^{(2)} = (w_{m/2+j}),\ \mathbf{z}^{(even)} = (z_{2j}),\ \mathbf{z}^{(odd)} = (z_{2j+1}) \in \mathbf{C}^{m/2} \end{array} \tag{14}$$

then

$$\begin{cases} \mathbf{z}^{(even)} = \mathrm{DFT}_{m/2}(\mathbf{w}^{(1)} + \mathbf{w}^{(2)}) \\ \mathbf{z}^{(odd)} = \mathrm{DFT}_{m/2}(D(\mathbf{w}^{(1)} - \mathbf{w}^{(2)})) \end{cases} \tag{15}$$

and

$$\begin{cases} \mathbf{w}^{(1)} = (\ \mathrm{IDFT}_{m/2}(\mathbf{z}^{(even)}) + \bar{D}\ \mathrm{IDFT}_{m/2}(\mathbf{z}^{(odd)}))/2 \\ \mathbf{w}^{(2)} = (\ \mathrm{IDFT}_{m/2}(\mathbf{z}^{(even)}) - \bar{D}\ \mathrm{IDFT}_{m/2}(\mathbf{z}^{(odd)}))/2 \end{cases} \tag{16}$$

where $\bar{D}$ is the complex conjugate of D.

The following problem, which consists in computing the DFT of a real vector of m components, is solved by means of the computation of a complex DFT of order $m/2$.

Problem 1. *Given the integer $m = 2^q$, $q > 0$ integer, and the real coefficients $a_0, a_1, \ldots, a_{m-1}$ of the polynomial $a(x) = \sum_{j=0}^{m-1} a_j x^j$, compute the values $b_j = a(\omega_m^j)$, $j = 0, \ldots, m-1$ that the polynomial $a(x)$ takes on at the m-th roots of 1.*

Solution. The following well-known formulae reduce Problem 1 to computing a complex DFT of order $m/2$. Let

$$\begin{array}{l} \mathbf{w} = (w_j), w_j = a_{2j} + \mathbf{i}a_{2j+1},\ j = 0, \ldots, m/2-1, \\ \mathbf{z} = \mathrm{DFT}_{m/2}(\mathbf{w}), \end{array} \tag{17}$$

then

$$\begin{array}{l} b_0 = \mathrm{re}(z_0) + \mathrm{im}(z_0),\ b_{m/2} = \mathrm{re}(z_0) - \mathrm{im}(z_0), \\ b_j = ((z_j + \bar{z}_{m/2-j}) - \mathbf{i}\omega_m^j(z_j - \bar{z}_{m/2-j}))/2, \\ b_{m/2+j} = \bar{b}_{m/2-j},\ j = 1, \ldots, m/2-1. \end{array} \tag{18}$$

□

The converse of Problem 1 is described below.

Problem 2. *Given the integer $m = 2^q$ as in Problem 1, and the values $b_j = a(\omega_m^j)$, $j = 0, \ldots, m-1$ that the real polynomial $a(x) = \sum_{j=0}^{m-1} a_j x^j$ takes on at the m-th roots of 1, compute the coefficients $a_0, a_1, \ldots, a_{m-1}$ of $a(x)$.*

Solution. Problem 2 can be easily reduced to the computation of a complex IDFT of order $m/2$; in fact, we have the following formulae that can be obtained by reverting (17) and (18). Let

$$\begin{array}{l} \mathbf{z} = (z_j), \\ z_j = ((b_j + b_{m/2+j}) + \mathbf{i}\bar{\omega}_m^j(b_j - b_{m/2+j}))/2,\ j = 0, \ldots, m/2-1, \\ \mathbf{w} = \mathrm{IDFT}_{m/2}(\mathbf{z}), \end{array} \tag{19}$$

then

$$a_{2j} = \mathrm{re}(w_j),\ a_{2j+1} = \mathrm{im}(w_j),\ j = 0, \ldots, m/2-1. \tag{20}$$

□

The next problem consists in computing the values that a polynomial of degree less than m takes on at the odd powers of ω_{2m}, once the values taken on at the even powers of ω_{2m} have been computed.

Problem 3. *Given the integer $m = 2^q$, as in Problem 1, the coefficients $a_0, a_1, \ldots, a_{m-1}$ of the polynomial $a(x)$ and the values $b_{2j} = a(\omega_{2m}^{2j}) = a(\omega_m^j)$, $j = 0, \ldots, m-1$ that $a(x)$ takes on at the even powers of ω_{2m}, compute the values $b_{2j+1} = a(\omega_{2m}^{2j+1})$, $j = 0, \ldots, m-1$, that $a(x)$ takes on at the odd powers of ω_{2m}.*

Solution. Indeed, Problem 3 can be solved by computing the DFT of order $2m$ of the real vector $\mathbf{u} = (\mathbf{u}^{(1)T}, \mathbf{u}^{(2)T})^T$, where $\mathbf{u}^{(1)} = (a_0, \ldots, a_{m-1})^T$, $\mathbf{u}^{(2)} = \mathbf{0}$, i.e., $\mathbf{v} = \mathrm{DFT}_{2m}(\mathbf{u})$. But in this way the values b_{2j} would be computed once again. In a more efficient way this problem can be solved by reducing it to the computation of a complex DFT of order $m/2$. In fact, by using (17) and (18) where m is replaced by $2m$, we find that Problem 3 is reduced to computing $\mathbf{z} = \mathrm{DFT}_m(\mathbf{w})$ where the vector $\mathbf{w} = (w_j)$ is such that $w_j = a_{2j} + \mathbf{i}a_{2j+1}$, $w_{m/2+j} = 0$, $j = 0, \ldots, m/2-1$. Partitionig $\mathbf{w}$ as ($\mathbf{w} = (\mathbf{w}^{(1)T}, \mathbf{w}^{(2)T})^T$ and applying (14) and (15) yields

$$\begin{array}{l} \mathbf{z}^{(even)} = \mathrm{DFT}_{m/2}(\mathbf{w}^{(1)}) \\ \mathbf{z}^{(odd)} = \mathrm{DFT}_{m/2}(D\mathbf{w}^{(1)}) \end{array}$$

since $\mathbf{w}^{(2)} = \mathbf{0}$. Now, again from (18) we find that the components b_{2j+1}, are fully defined by the components z_{2j+1}. In this way the computation is reduced to a single DFT of order $m/2$, namely $\mathrm{DFT}_{m/2}(D\mathbf{w}^{(1)})$. □

In the computation of the coefficients of the power series $\varphi^{(n+1)}(z)$, $\widehat{\varphi}^{(n+1)}(z)$ (7) by means of the evaluation/interpolation technique, we also need to solve the following problem.

Problem 4. *Given the integer $m = 2^q$ as in Problem 1, the values $b_{2j+1} = a(\omega_{2m}^{2j+1})$, $j = 0, \ldots, m-1$ that the real polynomial $a(x) = \sum_{j=0}^{2m-1} a_j x^j$ takes on at the odd powers of ω_{2m}, and $\mathbf{a}^{(1)} = \mathrm{IDFT}_m(\mathbf{b}^{(even)})$, $\mathbf{b}^{(even)} = (b_{2j})$, $b_{2j} = a(\omega_{2m}^{2j}) = a(\omega_m^j)$, $j = 0, \ldots, m-1$, compute the coefficients $a_0, a_1, \ldots, a_{2m-1}$ of $a(x)$.*

Solution. Indeed, this problem can be solved by computing $\mathbf{a} = \mathrm{IDFT}_{2m}\mathbf{b}$. A cheaper solution can be obtained as follows. By applying (19) and (20) where m is replaced by $2m$, we find that the vector $\mathbf{a}$ is recovered from the components of $\mathbf{w}$, where $\mathbf{w} = \mathrm{IDFT}_m(\mathbf{z})$. Now, in order to compute $\mathbf{w}$ we apply (14), (16) and reduce the problem to computing $\mathrm{IDFT}_{m/2}(\mathbf{z}^{(even)})$ and $\mathrm{IDFT}_{m/2}(\mathbf{z}^{(odd)})$. Now, the vector $\mathrm{IDFT}_{m/2}(\mathbf{z}^{(even)})$ is already available and needs the computation of no IDFT. In fact, observe that applying (19) and (20) for the computation of $\mathbf{a}^{(1)} = \mathrm{IDFT}_{m/2}(\mathbf{b}^{(even)})$, i.e., with a_j and b_j replaced with $a_j^{(1)}$ and b_{2j}, respectively, yields $\mathrm{IDFT}_{m/2}(\mathbf{z}^{(even)})$ as explicit function of $a_j^{(1)}$, more precisely, $\mathrm{IDFT}(\mathbf{z}^{(even)}) = (a_{2j}^{(1)} + \mathbf{i}a_{2j+1}^{(1)})$. □

5 Implementation of the algorithm and numerical results

By using the results presented in Sections 3 and 4 we may now give a more accurate description of Algorithm 2.1 in the form of the following:

Algorithm 5.1. *Computation of the solution G of (3).*

Input. Positive integers q_0, M_0, k, $M_0 = 2^{q_0}$, an error bound $\epsilon > 0$, and the nonnegative $k \times k$ matrices A_i, $i = 0, 1, \ldots, M_0$, such that the matrix $\sum_{i=0}^{M_0} A_i$ is ϵ-stochastic and the associated Markov chain is positive recurrent.

Output. An approximation $\widetilde{G}$ of the solution G of (3) together with an error bound δ such that $\mathbf{e}^T|\sum_{i=0}^{+\infty} \widetilde{G}^i A_i - \widetilde{G}| \le \delta \mathbf{e}^T$.

Computation. 1. *Initialization.*
Let $\varphi^{(0)}(z) = \sum_{i=0}^{M_0} A_i z^i$, $\widehat{\varphi}^{(0)}(z) = \sum_{i=0}^{M_0-1} A_{i+1} z^i$, $R^{(0)} = A_0(I - \sum_{i=1}^{M_0} A_i)^{-1}$, $n = 0$.

2. *Computation of the coefficients of the matrix polynomials $P(z)$ and $\widehat{P}(z)$ of degree M_n and $M_n - 1$ which approximate the matrix power series $\varphi^{(n)}(z)$ and $\widehat{\varphi}^{(n)}(z)$.*

   Repeat

   (a) Set $n = n + 1$, $q_n = q_{n-1} - 1$, $M_n = 2^{q_n}$.

   (b) Compute the mean values $\boldsymbol{\alpha}^{(n)}$ and $\widehat{\boldsymbol{\alpha}}^{(n)}$ by means of (11).

   (c) Evaluate the functions $\varphi_{odd}^{(n-1)}(z)$, $\varphi_{even}^{(n-1)}(z)$ and $\widehat{\varphi}_{odd}^{(n-1)}(z)$, $\widehat{\varphi}_{even}^{(n-1)}(z)$ at the set of Fourier points

   $$\mathcal{F}_{M_n} = \{\omega_{M_n}^i, \quad i = 0, 1, \ldots, M_n - 1\}.$$

   This computation is performed by means of the solution of Problem 1, if $M_n = 2^{q_{n-1}-1}$, that is, when this evaluation stage is encountered for the first time. The computation is performed by means of the solution of Problem 3, otherwise.

   (d) Pointwise apply equations (7), that is, successively compute the matrices

   $$\begin{aligned} S_1(z) &= \varphi_{even}^{(n-1)}(z)(I - \varphi_{odd}^{(n-1)}(z))^{-1} \\ S_2(z) &= z\varphi_{odd}^{(n-1)}(z) + S_1(z)\varphi_{even}^{(n-1)}(z) \\ \widehat{S}_2(z) &= \widehat{\varphi}_{odd}^{(n-1)}(z) + S_1(z)\widehat{\varphi}_{even}^{(n-1)}(z) \end{aligned} \tag{21}$$

   for $z \in \mathcal{F}_{M_n}$ if $M_n = 2^{q_{n-1}-1}$, that is, when this stage is encountered for the first time; otherwise for $z \in \mathcal{F}_{M_n} - \mathcal{F}_{M_n/2}$.

   (e) Interpolate the values of $S_2(z)$ and $\widehat{S}_2(z)$ at $z \in \mathcal{F}_{M_n}$, and obtain the coefficients of the matrix polynomials $P(z)$ and $\widehat{P}(z)$. This computation is performed by means of the solution of Problem 2, if $M_n = 2^{q_{n-1}-1}$, that is, when this interpolation stage is encountered for the first time. The computation is performed by means of the solution of Problem 4, otherwise.

   (f) Apply tests (12) and (13) in order to check if $P(z)$ and $\widehat{P}(z)$ are good approximations of the series. If the inequalities (12) and (13) are satisfied then skip to the next stage. Otherwise, set $M_n = 2M_n$, $q_n = q_n + 1$ and repeat from stage (2c).

   (g) Set $\varphi^{(n)}(z) = P(z)$, $\widehat{\varphi}^{(n)}(z) = \widehat{P}(z)$.

   (h) Compute $R^{(n)} = A_0^{(n)}(I - \sum_{i=1}^{M_n} A_i^{(n)})^{-1}$.

   Until one of the following conditions is verified.

   $$|R^{(n)} - R^{(n-1)}| < \epsilon E, \tag{C1}$$
   $$\mathbf{e}^T(I - A_0^{(n)}(I - A_1^{(n)})^{-1}) < \epsilon \mathbf{e}^T, \tag{C2}$$
   $$\mathbf{e}^T(I - A_0(I - \widehat{A}_1^{(n)})^{-1}) < \epsilon \mathbf{e}^T. \tag{C3}$$

3. *Computation of the matrix* $\widetilde{G}$.
   (a) If condition (C1) is verified then $\widetilde{G} = A_0(I - \sum_{i=0}^{\widehat{M}_n - 1} {R^{(n)}}^i \widehat{A}_{i+1}^{(n)})^{-1}$.
   (b) If condition (C2) is verified then $\widetilde{G} = A_0(I - \widehat{A}_1^{(n)})^{-1}$.
   (c) If condition (C3) is verified then $\widetilde{G} = A_0(I - \widehat{A}_1^{(n)})^{-1}$.
4. *Computation of the error bound* δ.
   Compute $\delta = \max_{n=1,\ldots,k} \sum_{i=1}^{k} |w_{i,j}|$, for $W = (w_{i,j})_{i,j}$, $W = \sum_{i=0}^{M_0} \widetilde{G}^i A_i - \widetilde{G}$.

The above algorithm has been implemented in Fortran90 in the subroutine `pwcr` (pointwise cyclic reduction). In order to use the subroutine `pwcr`, a driver program `pwcr_drv` is also provided. This program `pwcr_drv` reads from the standard input the file name `<input_file_name>` of the input data. This file must contain the folling data on each line:

1. On the first line: the block dimension k;
2. On the second line: the number $M_0 + 1$ of blocks A_i such that $\sum_{i=0}^{M_0} A_i$ is numerically stochastic;
3. On the third line: the error bound ϵ;
4. On the subsequent lines: the nonzero entries $(A_i)_{r,s}$ of the blocks A_i coded as follows: $i+1$, r, s, $(A_i)_{r,s}$;
5. On the last line: the quartuple $0, 0, 0, 0.d0$, which denotes the end of the data file.

The driver calls the subroutine `pwcr` and writes the output into the file `<input_file_name>.out`. The output file contains the residual error $|| \sum_{i=0}^{M_0} \widetilde{G}^i A_i - \widetilde{G}||_1$, where $\widetilde{G}$ is the computed approximation of the solution G of (3), and the entries $\widetilde{g}_{r,s}$ of $\widetilde{G}$ arranged row-wise.

The subroutine `pwcr` has the call sequence

```
CALL pwcr(a,eps,g,err)
```

where: `a` is a real(kind(0.d0)), dimension(:,:,:), pointer, containing the entries of the blocks A_i; `eps` is a real(kind(0.d0)) containing the error bound ϵ; `g` is a real(kind(0.d0)), dimension(:,:), pointer, containing the entries of the computed approximation $\widetilde{G}$; `err` is the residual error $|| \sum_{i=0}^{M_0} \widetilde{G}^i A_i - \widetilde{G}||_1$.

The main subroutine used inside `pwcr` is the subroutine `schur` which performs one step of cyclic reduction by computing the even and the odd components of the matrix series $\varphi^{(n+1)}(z)$ and $\widehat{\varphi}^{(n+1)}(z)$ (compare (7)) which define the Schur complement in the cyclic reduction process.

In order to compute the values of the series $\varphi^{(n+1)}(z)$ and $\widehat{\varphi}^{(n+1)}(z)$ at the roots of 1, several subroutines which implement the solutions of problems 1,2,3,4, of section 4 in the case of matrix polynomials, have been introduced. More

specifically the subroutines `ftb1` and `iftb1` solve problems 1 and 2, respectively; the subroutines `ftb2` and `iftb2` solve problems 3 and 4, respectively.

For the computation of the FFT's of real vectors, the split radix algorithm of Dhuamel-Hollman, implemented by Sorensen et al [19], has been modified in order to avoid the recomputation of the Fourier points. The subroutines that are involved in this computation are: `fft1`, `ifft1`, `ffts1`, `iffts1`, `ffts2`, `iffts2`, `twiddle`, `itwiddle`. The Fourier points are computed by the subroutine `fillroots`.

The subroutines involved in the computation of the FFT's are contained in the file `pwcr_fft.f90`. The subroutines `pwcr, schur`, and the related subroutines, are contained in the file `pwcr_sub.f90`.

We have tested our algorithm on a problem arising from the mathematical modeling of a Metaring MAC Protocol [1]. For this particular example the blocks A_i have dimension $k = 16$. We have tested the algorithm for different values of the parameter ρ, $0 < \rho < 1$, which represents the stability condition of the associated Markov chain [16]: as ρ tends to one, the problem of computing the solution G of (3) by using customary techniques becomes more difficult; in fact, for $\rho = 1$ the Markov chain is not positive recurrent [16]. The numerical degree of the series $\varphi^{(0)}(z)$ is equal to 168 for $\rho = 0.1$, is equal to 240 for $\rho = 0.8$, and is equal to 264 for the remaining tested values of $\rho \geq 0.9$. We have compared our algorithm with the cyclic reduction of [3] and with the functional iteration method based on the recursion

$$\begin{aligned} &X_0 = I, \\ &X_{n+1} = A_0 \left(I - \textstyle\sum_{i=2}^{+\infty} X_n^{i-1} A_i \right)^{-1}, \ n \geq 0, \end{aligned} \tag{22}$$

which is the fastest among the classical linearly convergent functional iterations [15].

The numerical experiments have been performed on an alpha workstation with a base 2 arithmetic endowed with 53 bits.

Table 1 reports the CPU time (in seconds) and the number of iterations needed by the Point-Wise Cyclic Reduction (PWCR) algorithm to compute an approximation of the matrix G, by choosing $\epsilon = 10^{-12}$; the residual of the computed approximation is also reported.

Table 2 reports the CPU time (in seconds) and the number of iterations needed by the Cyclic Reduction (CR) implemented in [3], the residual of the computed approximation, and the ratio between the times needed by CR and by PWCR. Observe that the algorithm PWCR, besides being faster than CR, leads to a higher accuracy of the result.

We also compared our method with classical techiques based on functional iterations: Table 3 reports the CPU time (in seconds) and the number of iterations needed by the Functional Iteration Formula (FIF) defined in (22), the residual of the computed approximation, and the ratio between the times needed by this algorithm and by PWCR.

ρ	Time (s.)	Iterations	Residual
0.1	0.9	9	$1.8 \cdot 10^{-13}$
0.8	1.5	13	$5.4 \cdot 10^{-14}$
0.9	2.3	14	$1.5 \cdot 10^{-14}$
0.95	2.4	16	$1.8 \cdot 10^{-14}$
0.96	2.4	17	$2.3 \cdot 10^{-14}$
0.97	2.5	20	$4.2 \cdot 10^{-14}$

Table 1: Point-Wise Cyclic Reduction

ρ	Time (s.)	Iterations	Residual	CCR/PWCR
0.1	1.3	9	$8.4 \cdot 10^{-14}$	1.4
0.8	3.2	13	$2.7 \cdot 10^{-13}$	2.1
0.9	3.3	14	$2.7 \cdot 10^{-13}$	1.4
0.95	3.4	16	$1.8 \cdot 10^{-13}$	1.4
0.96	3.4	17	$2.0 \cdot 10^{-13}$	1.4
0.97	3.5	20	$2.3 \cdot 10^{-13}$	1.4

Table 2: Customary Cyclic Reduction

Table 3 shows the effectiveness of our method, specially in cases where customary techniques converge very slowly. Indeed, observe that the cyclic reduction algorithm is almost insensitive to the values of the parameter ρ, whereas the performance of the functional iteration method strongly deteriorates as ρ approaches to 1.

References

[1] G. Anastasi, L. Lenzini, B. Meini, "Performance evaluation of a worst case model of the Metaring MAC Protocol with global fairness", *Performance Evaluation,* to appear.

[2] D. Bini, B. Meini, "On cyclic reduction applied to a class of Toeplitz-like matrices arising in queueing problems", *Computations with Markov Chains*, W. J. Stewart Ed., pp. 21–38, Kluwer Academic Publishers, 1996.

[3] D. Bini, B. Meini, "On the solution of a nonlinear matrix equation arising in queueing problems", *SIAM J. Matrix Anal. Appl.,* 17 (1996), pp. 906–926.

ρ	Time (s.)	Iterations	Residual	FIF/PWCR
0.1	0.3	22	$2.2 \cdot 10^{-14}$	0.3
0.8	3.9	148	$1.1 \cdot 10^{-13}$	2.6
0.9	10.8	373	$1.2 \cdot 10^{-13}$	4.7
0.95	44.4	1534	$1.3 \cdot 10^{-13}$	18.5
0.96	91.4	3158	$1.3 \cdot 10^{-13}$	38.1
0.97	96.8	3336	$1.3 \cdot 10^{-13}$	38.7

Table 3: Functional Iteration Method

[4] D. Bini, V. Pan, *Matrix and Polynomial Computations, Vol. 1: Fundamental Algorithms,* Birkhäuser, Boston 1994.

[5] D. Bini, V. Pan, "Polynomial division and its computational complexity", *J. Complexity,* 6 (1986), pp. 179–203.

[6] E. O. Brighman, *The Fast Fourier Transform*, Prentice Hall, Englewood Cliffs, N. J., 1974.

[7] B. L. Buzbee, G. H. Golub, C. W. Nielson, "On direct methods for solving Poisson's equation", *SIAM J. Numer. Anal.,* 7 (1970), pp. 627–656.

[8] E. Çinlar, *Introduction to Stochastic Processes,* Prentice Hall, Englewood Cliffs, N.J., 1975.

[9] D.F. Elliott, K.R. Rao, *Fast Transform Algorithms, Analyses, Applications,* Academic Press, New York, 1982.

[10] G. Latouche, "Algorithms for evaluating the matrix G in Markov chains of PH/G/1 type", Bellcore Technical Report, 1992.

[11] G. Latouche, "Newton's iteration for non-linear equations in Markov chains", *IMA J. of Numerical Analysis,* 14 (1994), pp. 583–598.

[12] G. Latouche, V. Ramaswami, "A logarithmic reduction algorithm for quasi-birth-death processes", *J. Appl. Prob.,* 30 (1993), pp. 650–674.

[13] G. Latouche, G. W. Stewart, "Numerical methods for M/G/1 type queues", *Computations with Markov Chains*, W. J. Stewart Ed., pp. 571–581, Kluwer Academic Publishers, 1996.

[14] B. Meini, "An improved FFT-based version of the Ramaswami formula", *Comm. Stat. Stochastic Models, 13 (1997),* to appear.

[15] B. Meini, "New convergence results on functional iteration techniques for the numerical solution of M/G/1 type Markov chains", *Numer. Math.,* to appear.

[16] M.F. Neuts, *Structured Stochastic Matrices of M/G/1 Type and Their Applications,* Dekker Inc., New York, 1989.

[17] V. Ramaswami, "A stable recursion for the steady state vector in Markov chains of $M/G/1$ type", *Commun. Statist. Stochastic Models,* 4 (1988), pp. 183-188.

[18] V. Ramaswami, "Nonlinear matrix equations in applied probability - Solution techniques and open problems", *SIAM Review,* 30 (1988), pp. 256-263.

[19] M. Sorensen, M. Heideman, C. Burrus, "On computing the split-radix FFT", *IEEE Trans. Acoust. Signal Processing,* 34 (1986), pp. 152–156.

[20] G. W. Stewart, "On the solution of block Hessenberg systems", *Numer. Lin. Algebra with Appl.*, 2 (1995), pp. 287–296.