EDITED BOOK TITLE

University of Maryland College Park

Institute for Advanced Computer Studies TR–95–37
Department of Computer Science TR–3440

Numerical Methods for M/G/1 Type Queues[1]

Guy Latouche G. W. Stewart[2]

ABSTRACT

Queues of M/G/1 type give rise to infinite embedded Markov chains whose transition matrices are upper block Hessenberg. The traditional algorithms for solving these queues have involved the computation of an intermediate matrix G. Recently a recursive descent method for solving block Hessenberg systems has been proposed. In this paper we explore the interrelations of the two methods.

[2]This report is available by anonymous ftp from `thales.cs.umd.edu` in the directory `pub/reports`.

[2]Department of Computer Science and Institute for Advanced Computer Studies, University of Maryland, College Park, MD 20742.

1

NUMERICAL METHODS FOR M/G/1 TYPE QUEUES

Guy Latouche*, G. W. Stewart**

** Université Libre de Bruxelles*
Département d'Informatique
CP 212, Boulevard du Triomphe
1050 Bruxelles, Belgium

*** Department of Computer Science*
University of Maryland
College Park, MD 20742

ABSTRACT

Queues of M/G/1 type give rise to infinite embedded Markov chains whose transition matrices are upper block Hessenberg. The traditional algorithms for solving these queues have involved the computation of an intermediate matrix G. Recently a recursive descent method for solving block Hessenberg systems has been proposed. In this paper we explore the interrelations of the two methods.

1 INTRODUCTION

Queues of M/G/1 type (for background see Neuts [8]) give rise to embedded Markov chains whose transition matrices have the form

$$P = \begin{pmatrix} B_1 & B_2 & B_3 & B_4 & \cdots \\ A_0 & A_1 & A_2 & A_3 & \ddots \\ 0 & A_0 & A_1 & A_2 & \ddots \\ 0 & 0 & A_0 & A_1 & \ddots \\ \vdots & \ddots & \ddots & \ddots & \ddots \end{pmatrix}. \tag{1.1}$$

Here the blocks are of order m. We will represent the states by ordered pairs (k, ℓ) $(k = 0, 1, \ldots; \ell = 1, 2, \ldots, m)$. If these pairs are ordered lexicographically, then k numbers the blocks in the partition (1.1) and ℓ numbers the position within the blocks. We say that the state (k, ℓ) is at *level* k. In typical applications, the levels correspond to the number of customers in the queue.

The chain is derived by looking at the queue immediately after each customer has been served. Between these departures many customers can arrive, but at the end only one leaves. This gives P its block Hessenberg structure. The Toeplitz structure says that the levels, except for the first, are indistinguishable.

If the chain is recurrent, there is a unique vector $\boldsymbol{\pi}^{\mathrm{T}}$ whose components sum to one — we write $\boldsymbol{\pi}^{\mathrm{T}}\mathbf{e}=1$, where $\mathbf{e}$ is the vector whose components are one — such that $\boldsymbol{\pi}^{\mathrm{T}} P = \boldsymbol{\pi}^{\mathrm{T}}$. If we set

$$\boldsymbol{\pi}^{\mathrm{T}} = \begin{pmatrix} \pi_{(0,1)} & \pi_{(0,2)} & \cdots & \pi_{(0,m)} & \pi_{(1,1)} & \cdots \end{pmatrix}$$

then $\pi_{(k,\ell)}$ is the stationary probability of being at state ℓ within level k.

The dual case where the transition matrix has the lower Hessenberg form

$$P = \begin{pmatrix} B_1 & A_0 & 0 & 0 & \cdots \\ B_2 & A_1 & A_0 & 0 & \ddots \\ B_3 & A_2 & A_1 & A_0 & \ddots \\ B_4 & A_3 & A_2 & A_1 & \ddots \\ \vdots & \ddots & \ddots & \ddots & \ddots \end{pmatrix}, \tag{1.2}$$

is also of interest. These are called Markov chains of GI/M/1 type (see Neuts [7] for background). Such chains are derived by looking at a queue immediately before each arrival. Between two arrival epochs, many customers may depart; hence P is lower Hessenberg.

These Markov chains have been studied by many authors; we only cite here Neuts [7, 8], Ramaswami [9], Grassmann and Heyman [1], Latouche [5]. A number of iterative procedures have been proposed, usually requiring the determination of an auxiliary $m \times m$ matrix, called G for queues of M/G/1 type and R for queues of GI/M/1 type.

In contrast to these procedures, Stewart [10, 11] proposes to directly determine the vector $\boldsymbol{\pi}^{\mathrm{T}}$, by an approach based on a recursive descent method for solving block Hessenberg systems. It turns out that this method also produces the matrix G or the matrix R as a byproduct. Our goal is to show the relations between the two approaches.

In the next section we will introduce the matrix G along with an iterative method for computing it. In the following section we discuss the method for solving block Hessenberg matrices. We then briefly consider the case of the matrix R.

2 THE MATRIX G

We consider here matrices of the type (1.1), and we introduce two auxiliary matrix of order m, called G and U. They may be defined probabilistically as follows.

> The (i, j)-element of G is the probability that starting from state (k, i) $(k \geq 1)$ the chain will first appear at level $k-1$ in state $(k-1, j)$.

> The (i, j)-element of U is the probability that starting from state (k, i) $(k \geq 1)$ the chain will first appear at level k in state (k, j) before it appears at any lower level.

A necessary and sufficient condition for the chain to be recurrent is that G be stochastic. For if the ith row sum of G is less than one, there is a positive probability that starting from state (k, i) the chain will never arrive at level $k-1$.

It is shown in [2] that G and U are related as follows:

$$\begin{array}{ll} 1. & U = A_1 + A_2 G + A_3 G^2 + \cdots, \\ 2. & G = (I-U)^{-1} A_0. \end{array} \tag{1.3}$$

Hence G satisfies the equation

$$G = A_0 + A_1 G + A_2 G^2 + \cdots, \tag{1.4}$$

and indeed G is the smallest (componentwise) nonnegative solution of (1.4).

It is further shown in [9] that if we partition

$$\boldsymbol{\pi}^{\mathrm{T}} = (\boldsymbol{\pi}_0^{\mathrm{T}} \;\; \boldsymbol{\pi}_1^{\mathrm{T}} \;\; \boldsymbol{\pi}_2^{\mathrm{T}} \;\; \cdots)$$

then $\boldsymbol{\pi}_0^{\mathrm{T}}$ satisfies

$$\boldsymbol{\pi}_0^{\mathrm{T}} \hat{B}_0 = \boldsymbol{\pi}_0^{\mathrm{T}},$$

and given $\boldsymbol{\pi}_0^{\mathrm{T}}$ we can compute $\boldsymbol{\pi}_k^{\mathrm{T}}$ recursively in the form

$$\boldsymbol{\pi}_k^{\mathrm{T}} = \left(\boldsymbol{\pi}_0^{\mathrm{T}} \hat{B}_k + \sum_{i=1}^{k-1} \boldsymbol{\pi}_i^{\mathrm{T}} \hat{A}_{k-i+1} \right) (I - U)^{-1},$$

where

$$\hat{A}_k = \sum_{i \geq k} A_i G^{i-k}$$

and

$$\hat{B}_k = \sum_{i \geq k} B_i G^{i-k}.$$

The recursion for $\boldsymbol{\pi}_k^{\mathrm{T}}$ is homogeneous in $\boldsymbol{\pi}_0^{\mathrm{T}}$; hence one can solve it with an unnormalized $\boldsymbol{\pi}_0^{\mathrm{T}}$ and later normalize $\boldsymbol{\pi}^{\mathrm{T}}$ so that it sums to one. On the other hand, with additional calculations it is possible to normalize $\boldsymbol{\pi}_0^{\mathrm{T}}$ at once (see [6] for details). This algorithm is, in essence, Gaussian elimination; computing G corresponds to the forward elimination phase, computing $\boldsymbol{\pi}^{\mathrm{T}}$ as outlined above corresponds to the backward substitution phase.

These equations indicate one reason why it has long been of interest to determine the matrix G, which was seen as the key to the whole computational procedure. Several algorithms have been proposed, among which the following requires the least number of iterations [3]:

$$\begin{array}{ll} 1. & G_0 = 0, \\ 2. & U_{\nu+1} = A_1 + A_2 G_\nu + A_3 G_\nu^2 + \cdots, \quad \nu = 0, 1 \ldots, \\ 3. & G_{\nu+1} = (I - U_{\nu+1})^{-1} A_0, \quad \nu = 0, 1 \ldots. \end{array} \tag{1.5}$$

The two matrix sequences monotonically converge respectively to G and U.

Turning to practical matters, we first note that we cannot form the infinite sums required by the algorithm. However, since P is stochastic,

$$\sum_{\nu=0}^{\infty} A_\nu \mathbf{e} = \mathbf{e}, \tag{1.6}$$

and we must have $\lim_{\nu \to \infty} A_\nu = 0$. Thus there is an index M such that A_ν is negligible for $\nu > M$, and we can truncate the sums in (1.5). Let us call the resulting iterates $G_{M,\nu}$.

Since M is finite, we can then evaluate the expression (1.5.2) by nested multiplication. Specifically,

$$\begin{array}{ll} 1. & U_{M,\nu+1} := A_M, \\ 2. & U_{M,\nu+1} := U_{M,\nu+1}G_{M,\nu} + A_\mu, \quad \mu = M-1, M-2 \ldots 1. \end{array} \tag{1.7}$$

The iteration can take advantage of special structure in the matrix. First, if the A_k are sparse, then the work in forming the products in (1.5.2) can be reduced. Second, if A_0 is of less than full rank, then so is G, and we can take advantage of this fact as follows.

Let $\mathrm{rank}(A_0) = r$, and let

$$A_0 = VW$$

where V and W are $m \times r$ and $r \times m$ matrices. Then from (1.5.3)

$$G_\nu = [(I - U_\nu)^{-1}V]W \equiv SW,$$

and we can write (1.7.2) in the form

$$U_{M,\nu+1} := (U_{M,\nu+1}S)W + A_\mu.$$

If the rank of A_0 is small—specifically less than $m/2$—the computation of $U_{M,\nu+1}S$ followed by the computation of $(U_{M,\nu+1}S)W$ will be cheaper than the direct computation of the matrix product $U_{M,\nu+1}G_{M,\nu}$.

An alternative characterization of the matrix G will be useful in the sequel. Let us think of P as partitioned into four blocks:

$$P = \begin{pmatrix} B_1 & P_{0\bullet} \\ P_{\bullet 0} & P_{\bullet\bullet} \end{pmatrix}, \tag{1.8}$$

with

$$P_{0\bullet} = (B_2\ B_3\ B_4\ \cdots), \tag{1.9}$$

$$P_{\bullet 0} = \begin{pmatrix} A_0 \\ 0 \\ 0 \\ \vdots \end{pmatrix}, \tag{1.10}$$

$$P_{\bullet\bullet} = \begin{pmatrix} A_1 & A_2 & A_3 & \cdots \\ A_0 & A_1 & A_2 & \ddots \\ 0 & A_0 & A_1 & \ddots \\ \vdots & \ddots & \ddots & \ddots \end{pmatrix}. \tag{1.11}$$

In view of (1.6) and the irreducibility of P, the matrix $P_{\bullet\bullet}$ is properly substochastic. The matrix G satisfies the equation

$$Q_{\bullet\bullet} \begin{pmatrix} G \\ G^2 \\ G^3 \\ \vdots \end{pmatrix} = \begin{pmatrix} A_0 \\ 0 \\ 0 \\ \vdots \end{pmatrix}, \tag{1.12}$$

where $Q_{\bullet\bullet} = I - P_{\bullet\bullet}$. In fact each row of (1.12) is just (1.4) multiplied by a power of G. In order to prove that the only solution consists of powers of G, we use an argument similar to that of [4], Equation (13).

3 BLOCK HESSENBERG SYSTEMS

The approach taken in this section is based on the following observation. Let $P_{\bullet\bullet}$ and $Q_{\bullet\bullet}$ be defined as in the last section. Equation (1.12) implies that if we can solve block Hessenberg systems, we can compute G. One possibility is Gaussian elimination. In this paper, however, we consider an alternative that has certain advantages when the matrix of the system is sparse or when A_0 is of defective rank. The algorithm is quite general and does not depend on the block Toeplitz structure of $Q_{\bullet\bullet}$; however, to keep our notation consistent, we will continue to work with $P_{\bullet\bullet}$. Since we cannot work with infinite matrices, we will truncate $P_{\bullet\bullet}$ and $Q_{\bullet\bullet}$ to be of order N.

The algorithm is a recursive descent method for solving the linear system

$$Q_{\bullet\bullet}x = b.$$

Partition $Q_{\bullet\bullet}$ in the form

$$Q_{\bullet\bullet} = \begin{pmatrix} Q_{\mathrm{nw}} & Q_{\mathrm{ne}} \\ Q_{\mathrm{sw}} & Q_{\mathrm{se}} \end{pmatrix}.$$

Here the subscripts refer to points in the compass. In our application the order of $Q_{\bullet\bullet}$ will always be a power of 2, and the partitioning will be chosen to split

$Q_{\bullet\bullet}$ exactly in half. Note that $Q_{\rm sw}$ is an almost empty matrix with A_0 sitting in its upper right-hand corner.

Now consider the system

$$\hat{Q}\begin{pmatrix}\hat{x}_{\rm n}\\ \hat{x}_{\rm s}\end{pmatrix} \equiv \begin{pmatrix}Q_{\rm nw} & Q_{\rm ne}\\ 0 & Q_{\rm se}\end{pmatrix}\begin{pmatrix}\hat{x}_{\rm n}\\ \hat{x}_{\rm s}\end{pmatrix} = \begin{pmatrix}b_{\rm n}\\ b_{\rm s}\end{pmatrix}. \tag{1.13}$$

Assuming that we can solve block Hessenberg systems of half the size of the original, we can solve (1.13) by the following algorithm:

1. Solve $Q_{\rm se}\hat{x}_{\rm s} = b_{\rm s}$,
2. Solve $Q_{\rm nw}\hat{x}_{\rm n} = b_{\rm n} - Q_{\rm ne}\hat{x}_{\rm s}$.

We will now show how to patch together the solutions $\hat{x}_{\rm s}$ and $\hat{x}_{\rm n}$ to get the solution x of the original equation.

As in the last section, suppose A_0 has the factorization

$$A_0 = VW,$$

where V and W are of full rank r. Let $Q_{\rm nw}$ be of block order t, and set

$$E = \begin{array}{c} \\ \\ \\ t+1 \\ \\ \\ \end{array}\begin{pmatrix}0\\ \vdots\\ 0\\ V\\ \vdots\\ 0\end{pmatrix} \quad \text{and} \quad F = \begin{array}{c} \begin{array}{cccccc} & & t & & & \end{array} \\ \begin{pmatrix}0 & \cdots & W & 0 & \cdots & 0\end{pmatrix} \end{array}. \tag{1.14}$$

Set

$$\begin{array}{ll} 1. & X = F\hat{Q}^{-1},\\ 2. & S = I + F\hat{Q}^{-1}E,\\ 3. & Y = \hat{Q}^{-1}E. \end{array} \tag{1.15}$$

It can be shown that if we partition

$$\hat{x}^{\rm T} = (\hat{x}_1^{\rm T}\ \cdots\ \hat{x}_t^{\rm T}\ \hat{x}_{t+1}^{\rm T}\ \cdots\ \hat{x}_N^{\rm T})$$

then

$$x = \hat{x} - YS^{-1}\hat{x}_t.$$

An implementation of this method has been described in [11], and we will only point out some salient features. The matrices X and Y are called patch matrices. As we have seen, Y is used to solve systems of the form $Q_{\bullet\bullet}x = b$. The patch matrix X is used to solve transposed systems. Patch matrices must be precomputed, and they are analogous to the elimination phase of Gaussian elimination. It turns out that the bulk of the work, at least for dense matrices, consists of matrix vector multiplications of the form $Q_{\mathrm{ne}}x$. If Q_{ne} is sparse, operations can be saved in this process. Moreover, the number of columns in the patch matrix Y is equal to the rank of A_0, so that, as in the previous section, the smaller the rank, the more efficient the algorithm. Unlike the iteration in the previous section, we do not have to restrict the rank to be less than $m/2$.

The patch matrix Y contains the wherewithal to compute an approximation to G. To see this, partition

$$Y^{\mathrm{T}} = (Y_1^{\mathrm{T}} \ \cdots \ Y_t^{\mathrm{T}} \ Y_{t+1}^{\mathrm{T}} \ \cdots \ Y_N^{\mathrm{T}}).$$

Then it follows from (1.14) and (1.15.3) that

$$Q_{\mathrm{se}} \begin{pmatrix} Y_{t+1} \\ Y_{t+2} \\ \vdots \\ Y_N \end{pmatrix} = \begin{pmatrix} V \\ 0 \\ \vdots \\ 0 \end{pmatrix}. \tag{1.16}$$

On multiplying this equation from the right by W, we get

$$Q_{\mathrm{se}} \begin{pmatrix} Y_{t+1}W \\ Y_{t+2}W \\ \vdots \\ Y_N W \end{pmatrix} = \begin{pmatrix} VW \\ 0 \\ \vdots \\ 0 \end{pmatrix} = \begin{pmatrix} A_0 \\ 0 \\ \vdots \\ 0 \end{pmatrix}.$$

But this is just a truncated version of the equation (1.12), and it follows that

$$G \cong Y_{t+1}W.$$

Thus we have a new algorithm for computing G.

The particular form of our application generates some economies. In practice we would not only truncate the matrix $P_{\bullet\bullet}$ to be of block order N, but as we did in the last section, we throw away all A_k for k greater than some integer M. This gives our matrix a banded structure: for large N almost all the matrix Q_{ne} is zero, and we can take advantage of this sparsity.

Let us denote the patch matrices so obtained by $G_{M,N}$. Then a natural algorithm is to start with $G_{M,1}$ and successively compute $G_{M,2}$, $G_{M,4}$, $G_{M,8}$, and continue the doubling until convergence. Because our matrices are Toeplitz, $Q_{\mathrm{ne}} = Q_{sw}$, and the two matrices have the same patch matrices. This means that we can save work in passing from $G_{M,N}$ to $G_{M,2N}$. Moreover, because of the special nature of the system (1.16), we can compute $G_{M,N}$ very cheaply and check convergence before going on the the computation of the patch matrices for the doubled system.

Turning now to the computation of $\boldsymbol{\pi}^{\mathrm{T}}$, we will continue to assume that our system has been truncated. We begin by observing that

$$\boldsymbol{\pi}^{\mathrm{T}}(I - P) \equiv \boldsymbol{\pi}^{\mathrm{T}} Q = 0.$$

Writing this equation in partitioned form, we get

$$(\boldsymbol{\pi}_0^{\mathrm{T}} \ \boldsymbol{\pi}_\bullet^{\mathrm{T}}) \begin{pmatrix} Q_{00} & Q_{0\bullet} \\ Q_{\bullet 0} & Q_{\bullet\bullet} \end{pmatrix} = 0.$$

It then follows on eliminating $\boldsymbol{\pi}_\bullet^{\mathrm{T}}$ that

$$\boldsymbol{\pi}_0^{\mathrm{T}}(Q_{00} - Q_{0\bullet} Q_{\bullet\bullet}^{-1} Q_{\bullet 0}) = 0. \tag{1.17}$$

This is a small problem of order m which can be solved by standard techniques.

Now $Q_{\bullet 0}$ is the block vector whose first block is $-A_0$ and whose other blocks are zero. It follows that $Q_{\bullet\bullet}^{-1} Q_{\bullet 0} = -Y_\bullet W$, where $Y_\bullet$ is the block vector on the left of (1.16). Hence the system (1.17) has the form

$$\boldsymbol{\pi}_0^{\mathrm{T}}(Q_{00} + Q_{0\bullet} Y_\bullet W) = 0.$$

In particular for our truncated system

$$Q_{0\bullet} Y = -\sum_{i=1}^{N} B_i Y_i.$$

Finally, the vector $\boldsymbol{\pi}_\bullet^{\mathrm{T}}$ can be computed by solving the system

$$\boldsymbol{\pi}_\bullet^{\mathrm{T}} Q_{\bullet\bullet} = -\boldsymbol{\pi}_0^{\mathrm{T}} Q_{0\bullet}.$$

4 THE MATRIX R

We consider here matrices of the type (1.2). If we partition P as above, we have that

$$P_{0\bullet} \quad = \quad (A_0 \ 0 \ 0 \ \cdots), \tag{1.18}$$

$$
P_{\bullet 0} = \begin{pmatrix} B_2 \\ B_3 \\ B_4 \\ \vdots \end{pmatrix}, \tag{1.19}
$$

$$
P_{\bullet\bullet} = \begin{pmatrix} A_1 & A_0 & 0 & \cdots \\ A_2 & A_1 & A_0 & \ddots \\ A_3 & A_2 & A_1 & \ddots \\ \vdots & \ddots & \ddots & \ddots \end{pmatrix}. \tag{1.20}
$$

In this case, one proves [7] that there exists a matrix R of order m such that the stationary probability vector is given by

$$
\boldsymbol{\pi}_k^{\mathrm{T}} = \boldsymbol{\pi}_0^{\mathrm{T}} R^k, \qquad k \geq 0, \tag{1.21}
$$

and $\boldsymbol{\pi}_0^{\mathrm{T}}$ is a left eigenvector of the matrix $B[R] = \sum_{n \geq 1} B_n R^{n-1}$, normalized so that $\boldsymbol{\pi}_0^{\mathrm{T}}(I - R)^{-1}\mathrm{e} = 1$.

The matrix R is equal to $A_0(I - U)^{-1}$, where U is defined as in Section 2. As in the case of the matrix G, several algorithms have been proposed, in order to compute R, most of which are described and compared in [5]. The iteration corresponding to (1.5) is given below; the two sequences converge monotonically respectively to R and U.

$$
\begin{array}{ll}
1. & R_0 = 0, \\
2. & U_{\nu+1} = A_1 + R_\nu A_2 + R_\nu^2 A_3 + \cdots, \quad \nu = 0, 1 \ldots, \\
3. & R_{\nu+1} = A_0(I - U_{\nu+1})^{-1}, \quad \nu = 0, 1 \ldots.
\end{array} \tag{1.22}
$$

Of particular interest to us is the fact that the matrix R satisfies the following equation [4]:

$$
(R \; R^2 \; R^3 \; \ldots) Q_{\bullet\bullet} = (A_0 \; 0 \; 0 \; \ldots), \tag{1.23}
$$

where $Q_{\bullet\bullet} = (I - P_{\bullet\bullet})$. This clearly indicates that the matrix R may be determined by applying, mutatis mutandis, the recursive descent described in the preceding section.

Acknowledgements

This work was supported in part by the National Science Foundation under grant CCR 9115568.

REFERENCES

[1] W. Grassmann and D. P. Heyman. Equilibrium distribution of block-structured Markov chains with repeating rows. *J. Appl. Prob.*, 27:557–576, 1990.

[2] G. Latouche. A note on two matrices occurring in the solution of quasi-birth-and-death processes. *Commun. Statist. — Stochastic Models*, 3:251–257, 1987.

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

[4] G. Latouche. A simple proof for the matrix-geometric theorem. *Applied Stochastic Models and Data Analysis*, 8:25–29, 1992.

[5] G. Latouche. Algorithms for infinite Markov chains with repeating columns. In C. D. Meyer and R. J. Plemmons, editors, *Linear Algebra, Markov Chains, and Queueing Models*, pages 231–266, New York, 1993. Springer. The IMA Volumes in Mathematics and Its Applications, Volume 48.

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

[7] M. Neuts. *Matrix-Geometric Solutions in Stochastic Models.* Johns Hopkins University Press, Baltimore, 1981.

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

[9] V. Ramaswami. Stable recursion for the steady state vector for Markov chains of M/G/1 type. *Commun. Statist.—Stochastic Models*, 4:183–188, 1988.

[10] G. W. Stewart. On the solution of block hessenberg systems. Technical Report CS-TR-2973, Department of Computer Science, University of Maryland, College Park, 1992. To appear in *Numerical Linear Algebra and Applications.*.

[11] G. W. Stewart. Implementing an algorithm for solving block Hessenberg systems. Technical Report CS-TR-3295, Department of Computer Science, University of Maryland, 1994.