OVERCOMING INSTABILITY IN COMPUTING THE FUNDAMENTAL MATRIX FOR A MARKOV CHAIN

DANIEL P. HEYMAN* AND DIANNE P. O'LEARY†

April 5, 1996

Abstract. We present an algorithm for solving linear systems involving the probability or rate matrix for a Markov chain. It is based on a UL factorization but works only with a submatrix of the factor U. We demonstrate its utility on Erlang-B models as well as more complicated models of a telephone multiplexing system.

Key words. Markov chains, fundamental matrix, decision process.

1. Introduction. Markov chain models can lend insight into the behavior of many physical systems, such as telephone networks, highway systems, and ATM switching networks. These models are based on properties of a matrix P whose entries depend on the probabilities of transition from one state to another, or on the arrival and departure rates for customers. The matrix P is nonnegative. If we define D to be a diagonal matrix whose diagonal entries are the rowsums for P, then the matrix $D - P$ has zero rowsums. In other words,

$$(D - P)e = 0,$$

where e is the column vector of all ones. Thus, $D - P$ has a zero eigenvalue, and we denote its left eigenvector, normalized so that its entries sum to one, as the row vector π:

$$\pi(D - P) = 0^T \ , \quad \pi e = 1 \ .$$

The vector π gives information about the long-term behavior of the system; for example, if the entries in P are transition probabilities (so that $D = I$), then π is the stationary vector for the chain.

Systems analysts are interested in other computational quantities that give information about the short-term behavior of the chain. The *fundamental matrix* is defined to be

$$F = (D - P - e\pi)^{-1},$$

and the *group generalized inverse* is

$$A^{\#} \equiv F - e\pi.$$

(See, for example, [6].) The entries in these matrices are useful in computing mean first passage times, in computing biases in the entries in π as approximations to expected number of visits, and in determining decision rules to govern the control of the system. See [2] and [5] for some discussion of these applications.

The GTH algorithm [1] is an efficient algorithm for determining a factorization of the matrix $D - P$. From this factorization, all the other quantities can easily be computed.

* Bellcore, 331 Newman Springs Road, Red Bank, NJ 07701-7020 (dph@bellcore.com).

† Department of Computer Science and Institute for Advanced Computer Studies, University of Maryland, College Park, MD 20742. This work was supported by the National Science Foundation under Grant CCR-95-03126 (oleary@cs.umd.edu).

The GTH algorithm can be interpreted as a variant of Gauss-Elimination that differs from the usual LU form in two ways:

1. The elimination proceeds from the bottom of the matrix to the top (rather than top-to-bottom) and thus produces factors

$$D - P = UL,$$

where U is an upper triangular matrix with diagonal elements equal to -1, and L is a lower triangular matrix, with $l_{11} = 0$.

2. Since the rowsums of $D - P$ are zero, so are the rowsums of the matrix L. We compute the main diagonal elements of L to satisfy this constraint, rather than using the usual Gaussian elimination formulas. This modification has been shown to provide a very strong form of numerical stability when making use of these factors to compute the stationary vectors [7].

We assume that the rows of the matrix (one for each state of the chain) are numbered $\{0, 1, \ldots, n\}$, we let `P(i:j,k:l)` denote the submatrix of P consisting of elements in rows $i, i+1, \ldots, j$ and columns $k, k+1, \ldots, l$, and we let `sum(P(i,j:k))` denote the sum of the elements in row i and columns j through k. Then the GTH algorithm computes $UL = D - P$. It can be done using no additional matrix or vector storage as follows:

```
For i=n, n-1, ... , 1,
   s=sum(P(i,0:i-1))
   P(i,i)=-s
   P(0:i-1,i)=P(0:i-1,i)/s
   P(0:i-1,0:i-1)=P(0:i-1,0:i-1) + P(0:i-1,i)*P(i,0:i-1)
end for
P(0,0)=0
```

Then the matrix L is stored in the lower triangular part of P (including the main diagonal), and U is in the upper triangular, with main diagonal elements understood to be equal to -1. The entire factorization process takes $2/3\, n^3 + O(n^2)$ operations.

Once the UL factors of $D - P$ are determined, it is easy to compute the stationary vector from

$$\pi U = z,$$

where z is the first row of the identity matrix. Then the fundamental matrix can be computed from

$$ULF = I - e\pi,$$

with normalization $\pi F = \pi$, or the group generalized inverse from

$$ULA^{\#} = I - e\pi$$

with normalization $\pi A^{\#} = 0^T$.

Clearly, the lower triangular matrix L is singular, since its first row is zero. Conventional wisdom says that the matrix U is usually well-conditioned, but occasionally, this fails to be true, even if the nonzero singular values of $D - P$ are well behaved.

The purpose of this note is to exhibit examples of this phenomenon and to propose a more stable way to use the UL factors to compute the fundamental matrix and other quantities.

TABLE 2.1
Algorithm results on the examples of §2.

n	r_{orig}	$r_{improved}$	cond(U)	cond($D-P$)
5	4.2e-15	1.0e-15	2.4e+02	8.1e+00
10	3.6e-13	3.0e-15	5.4e+04	2.0e+01
15	1.3e-12	9.0e-15	1.1e+07	3.4e+01
20	4.4e-09	1.5e-14	1.9e+09	5.0e+01
25	1.3e-06	1.7e-14	3.3e+11	6.6e+01
30	2.5e-04	2.9e-14	5.6e+13	8.2e+01
35	2.8e-02	3.2e-14	9.2e+15	9.9e+01
40	2.5e+00	3.3e-14	1.5e+18	1.2e+02
45	4.1e+00	3.8e-14	Inf	1.3e+02
50	1.1e+05	5.5e-14	Inf	1.5e+02

2. A Troublesome Family of Examples. Consider an *Erlang-B model* of telephone traffic. Calls arrive as a Poisson process at rate λ to be served by n parallel servers at unit rate. A call that finds all servers busy is discarded.

This model yields a continuous-time Markov chain with states $\{0, 1, \ldots, n\}$. Let r_{ij} be the rate of passage from state i to state j. Then the only nonzero rates are

$$r_{i,i+1} = \lambda, \text{ for } 0 \leq i < n,$$
$$r_{i,i-1} = i, \text{ for } 0 < i \leq n.$$

Let the matrix $P = (r_{ij})$ and let D be the diagonal matrix whose entries are the rowsums of P.

Then it is easy to compute the UL factors of $D-P$: both U and L are bidiagonal matrices with nonzero entries

$$\begin{aligned} u_{ii} &= -1, \quad i = 0, \ldots, n, \\ u_{i,i+1} &= \frac{\lambda}{i+1}, \quad i = 0, \ldots, n-1, \\ l_{ii} &= -i, \quad i = 0, \ldots, n, \\ l_{i,i-1} &= i, \quad i = 1, \ldots, n. \end{aligned}$$

The unnormalized steady-state probabilities are

$$\pi_i = \frac{\lambda^i}{i!}, i = 0, \ldots, n.$$

If we set $\lambda = n$ and use this data to compute the last column of the fundamental matrix, we get the results in Table 1.

These results were computed using double-precision IEEE arithmetic (approximately 16 decimal digits) in Matlab.

The column labeled r_{orig} gives the norm of the residual vector when the last column of F is computed using the UL factors: i.e.,

$$r_{orig} = \|e_n - \pi_n e - (D-P) f_{orig}\|,$$

where e_n is the last column of the identity matrix and f_{orig} is the result of using forward and back substitution on the linear system

$$U L f_{orig} = e_n - \pi_n e.$$

We see that r_{orig} grows rapidly as n grows, although in exact arithmetic r_{orig} would be zero.

Such large residuals are a symptom of ill-conditioning, so the table also gives the condition number of U and the condition number of $D - P$. Here we define the condition number as the ratio of the largest to the smallest singular value of the matrix, although, since $D - P$ is singular, we leave out its zero singular value in computing this ratio. Clearly, the matrix U is rapidly approaching singularity, and thus when we use U to solve for the last column of F, accuracy can be lost.

In the next section we describe an improved algorithm that produces the residuals labeled $r_{improved}$ in the table.

3. A More Stable Way to use the UL Factors. To see what went wrong, we need to look at the null spaces of our various matrices.

Suppose we are solving $(D - P)z = b$, where b is in the range of $D - P$. Then the solution vector z satisfies

$$Lz = y,$$

where y is the solution to

$$Uy = b.$$

Since the top row of L is zero, the top element of y must also be zero in order for the system to have a solution. Thus, before we begin the back substitution on U, we already know the top component of y.

If, due to round-off error and ill-conditioning of U, the top component of the computed y fails to be close to zero, then our computation will not produce a good solution.

This insight also leads to a remedy. Instead of solving $Uy = b$, we can solve a system that involves only the last n components of y, knowing that the top one is zero. If we let $\bar{U}$ be the matrix formed by deleting the zeroth column of U, and let $\bar{y}$ be the vector formed by deleting the zeroth element of y, then we can compute $\bar{y}$ by solving the linear system

$$\bar{U}\bar{y} = b.$$

The matrix $\bar{U}$ is not upper triangular (In fact, it is zero *above* the main diagonal for the examples in §2, since U is bidiagonal). But $\bar{U}$ is always *upper Hessenberg*, with zeroes below the first subdiagonal. A sequence of n row operations reduces it to upper triangular form, at a cost of at most $O(n^2)$ floating point operations. Since the system is compatible, the same sequence of operations reduces the last component of b to zero, permitting back-substitution starting with equation $n - 1$.

We choose to reduce the matrix $\bar{U}$ to upper triangular form using the LU algorithm with partial pivoting. Just as in the GTH algorithm, only additions and divisions are being performed and no cancellation can occur, and the factorization can be done in-place, except for an auxiliary integer vector of permutation indices. In the following code fragment, we factor the matrix $\bar{U} = \tilde{L}\tilde{U}$, assuming that $\bar{U}$ is stored in the array P. We store $\tilde{U}$ in the upper triangle of P, rows 1 through n, and we store the multipliers (off-diagonal elements of the L factor) in the zeroth row of P. None of this disturbs the lower triangular factor L stored below the diagonal of P.

```
Initialize two row vectors of length  n+1:
  all entries in  ipos  are zero,
  and the i-th entry of  ind  is  i.

for i=1, ... , n,
  if |u(0,i)|  > 1,
    Interchange ind(0) with ind(i)
       and P(0,i:n) with P(i,i:n).
    Set ipos(i-1)=i.
  end if
  The pivot element is P(0,i)=-P(0,i)/P(i,i).
  Update row 0 as
    P(0,i+1:n)=P(0,i+1:n)+ P(0,i) *P(i,i+1:n).
end
```

This takes $n^2 + O(n)$ operations. The vector `ipos` is redundant, but included for clarity.

We use these factors as follows to solve the linear system $ULz = b$. First we solve $\tilde{L}q = b$ in $O(n)$ operations, by using the multipliers and the permutation information:

```
Let  q  be the vector  b  reordered
   as indicated by  ind.

for i=1, ... , n,
   Set ispot=ipos(i).
   Let q(ispot)=q(ispot)+P(0,i)*q(i).
end
```

Then we solve $\tilde{U}\bar{y} = q$ using backsubstitution. This takes $n^2 + O(n)$ operations.

Finally, we solve $Lz = \bar{y}$, setting $z_1 = 0$. This takes $n^2 + O(n)$ operations.

Applying this algorithm to the examples in §2 yields the results labeled $r_{improved}$ in Table 1. The improved algorithm yields a small residual for all of the examples. Since the residual norm divided by the condition number of $D-P$ is close to machine precision, we see that we have achieved attainable accuracy using this algorithm.

4. An Application. This work was motivated by difficulties encountered in computing solutions to the telecommunications model described in Krishnan and Huebner [5]. In their model, there are n channels that serve C classes of calls. Class-j calls arrive according to a Poisson process at rate λ_j, have exponential service times with mean $1/\mu_j$, and each call requires r_j channels. The classes represent different types of applications, such as voice, data and video. The problem is to construct an admission rule that optimizes a given performance criterion, e.g., minimize the loss rate of calls. To illustrate the nature of the problem, suppose $r_1 > r_2$, and exactly r_1 channels are free when a class-1 job arrives. Accepting this job may preclude accepting several class-2 calls that might arrive soon. The class-1 job should be accepted when μ_1 is sufficiently large and λ_2 is sufficiently small so that the expected number of lost calls is less than one. This expected value depends on which calls are currently in progress (because some of them may finish soon enough to allow some class-2 calls to be admitted in the near future) as well as on which type of call is under consideration.

Krishnan and Huebner formulate this problem as a Markov decision process. This involves constructing a Markov chain to model the number of occupied channels at any time, so there is an underlying continuous-time Markov chain with states $\{0, 1, \ldots, n\}$. The nonzero elements in the rate matrix P for this chain are defined by

$$\begin{array}{lll} p_{i,i+r_k} = \lambda_k & \text{for } 0 \le i \le C - r_k, & k = 1, \ldots, C, \\ p_{i,i-r_k} = \mu_k E(m_k|i) = \frac{\lambda_k q(i-r_k)}{q(i)} & \text{for } r_k \le i \le n, & k = 1, \ldots, C. \end{array}$$

The state probabilities $q(i)$ are computed recursively using a method of Kaufman [4].

The examples of §2 are special cases of this model with $C = 1$ class.

Let c_j be the "cost" per unit time of being in state j; e.g., the loss rate if the objective is to minimize the loss rate of calls. This model is described in continuous time, but it can be converted into a discrete-time model where transitions occur at times $1, 2, \ldots$ by "uniformizing" the model; see, e.g., Heyman and Sobel [3, §8-7] for details. Let P be the transition matrix of the discrete-time Markov chain; P inherits the state space $\{0, 1, 2, \ldots, n\}$ and has elements p_{ij}. If some $r_j = 1$, then P is irreducible and aperiodic and has no transient states. Otherwise, some states may not be reachable (e.g. state 1 when starting empty); these states should be eliminated.

Krishnan and Huebner show that when i channels are occupied and a class-j call arrives, that call should be admitted if and only if,

$$t(i) < t(i + r_j)$$

where

$$t(k) = \sum_{j=0}^{n} A^{\#}_{kj} c_j, \quad k = 0, 1, \ldots, n\,.$$

From this equation (a variant of the one used by Krishnan and Huebner) we see that we need to compute the jth column of $A^{\#}$ when $c_j \neq 0$.

Example: Suppose we have $n = 100$ trunk lines, with $C = 3$ classes of traffic defined by mean arrival times (λ_i), mean holding time $(1/\mu_i)$, and r_i trunks required per call as follows:

i	λ_i	μ_i	r_i
1	20	1	1
2	20	1/2	2
3	5	1/3	3

Using the standard algorithm, we obtain a residual of size 9.4e+12 for $j = n$. The condition number of U is computationally infinite, even though the condition number of $D - P$ is only 83. Using the algorithm from §3, however, we obtain a residual of size 9.6e-14.

5. Conclusions. We have presented an improvement to algorithms that use the UL factors to compute quantities related to Markov chains. For a dense matrix, it requires only $O(n^2)$ additional operations compared to the standard $O(n^3)$ algorithm but improves the accuracy obtained in the results. The same approach could be used on sparse matrices arising from Markov chains.

REFERENCES

[1] W. K. Grassmann, M. I. Taksar, and D. P. Heyman, *Regenerative analysis and steady state distributions*, Operations Research, 33 (1985), pp. 1107–1116.

[2] D. P. Heyman and D. P. O'Leary, *What is fundamental for Markov chains: First passage times, fundamental matrices, and group generalized inverses*, in Proceedings of the Second International Workshop on Markov Chains, W. Stewart, ed., Kluwer Academic Publishers, 1995, pp. 151–161.

[3] D. P. Heyman and M. J. Sobel, *Stochastic Models in Operations Research, vol. I*, McGraw-Hill, New York, 1982.

[4] J. S. Kaufman, *Blocking in a shared resource environment*, IEEE Trans. on Communications, COM-29 (1981), pp. 1474–1481.

[5] K. R. Krishnan and F. Huebner, *Admission control for multirate CBR traffic: A Markov decision criterion*, tech. report, Bellcore, Morristown, NJ, 1996 preprint.

[6] C. D. Meyer, Jr., *The role of the group generalized inverse in the theory of Markov chains*, SIAM Review, 17 (1975), pp. 443–464.

[7] C. A. O'Cinneide, *Entrywise perturbation theory and error analysis for Markov chains*, Numerische Mathematik, 65 (1993), pp. 109–120.