



A direct projection method for Markov chains

Michele Benzi¹

Department of Mathematics and Computer Science, Emory University, Atlanta, GA 30322, USA

Received 12 August 2003; accepted 3 December 2003

Submitted by D. Szyld

Dedicated to Carl Meyer on his 60th birthday

Abstract

A direct method based on oblique projections is adapted to compute the stationary distribution vector of a finite Markov chain. The algorithm can also be used to compute the group inverse of the corresponding generator matrix. It is shown how to update the stationary vector and other quantities of interest when one row of the transition probability matrix is modified. A GTH-like variant that appears to compute the stationary probabilities to high relative accuracy is developed.

© 2004 Elsevier Inc. All rights reserved.

Keywords: Finite Markov chains; Stationary vector; Projections; Direct solution method; Purcell's method; Gaussian elimination; Generalized inverses; Rank-one updates; Nearly uncoupled chains; GTH algorithm

1. Introduction

This paper is concerned with direct solution methods for computing stationary vectors and group inverses associated with finite Markov chains. The stationary probability distribution vector of a discrete-time, ergodic Markov process with $n \times n$ (row-stochastic) transition probability matrix $P = [p_{ij}]$ is the unique row vector $\pi = [\pi_1, \pi_2, \dots, \pi_n]$ which satisfies

$$\pi = \pi P, \quad \pi_i > 0, \quad \sum_{i=1}^n \pi_i = 1. \quad (1)$$

E-mail address: benzi@mathcs.emory.edu (M. Benzi).

¹ Part of this author's work was supported by NSF grant DMS-0207599.

Letting $x = \pi^T$ and $A = I_n - P^T$, where I_n denotes the $n \times n$ identity matrix, the computation of the stationary vector reduces to finding a nontrivial solution to the homogeneous linear system $Ax = 0$. The ergodicity assumption means that P (and therefore A) is irreducible. Perron–Frobenius theory [7] guarantees that A has rank $n - 1$, and that the (one-dimensional) null space $\mathcal{N}(A)$ of A is spanned by a vector x with positive entries. Upon normalization in the ℓ_1 -norm, this is the stationary distribution vector of the Markov process. The (unsymmetric) coefficient matrix A is a singular M -matrix, called the *generator* of the Markov process. (Strictly speaking, the generator matrix is $Q = P - I_n = -A^T$. Here A is used instead of Q in order to conform to the familiar notation of numerical linear algebra.)

A number of methods, both direct and iterative, for solving (1) are surveyed by Stewart in the excellent monograph [41]. Direct solution methods for Markov chains include, besides Gaussian elimination, its variant known as the Grassmann–Taksar–Heyman (GTH) algorithm [22], which is guaranteed to compute the stationary vector with low relative error in each component, and the QR factorization [21].

In this paper, a direct method based on oblique projections is considered. This elegant algorithm, which is closely related to LU factorization, is sometimes attributed to Purcell, who presented it in a short paper [37] as an alternative to Gaussian elimination for solving nonsingular systems $Ax = b$. See also the brief descriptions in [13, pp. 171–173; 25, p. 142]. The original paper did not address numerical stability issues and did not give a geometric interpretation of the procedure. The basic idea of the algorithm has been rediscovered by several authors over the years; see the historical notes in [2,14] and the remarks in the next section. In spite of the favorable review [15] by no one less than George Forsythe, who called it a “promising new method”, Purcell’s algorithm is still largely unknown even among numerical linear algebra experts. Whatever the causes for this situation, Purcell’s method has recently seen renewed interest. A variant for general sparse matrices has been developed in [2,3]. A round-off error analysis of this algorithm for a general (nonsingular) matrix has been carried out by Fletcher in [14], where it is also shown that the method has certain advantages over Gaussian elimination in linear programming problems. Recently, the potential of Purcell’s method as a solver for dense systems on parallel computer architectures has been demonstrated in [9].

The purpose of this paper is to explain how this algorithm can be used to solve singular Markov-type systems, and to illustrate some of its features in such context. It is also shown how to use the algorithm to compute various types of generalized inverses of the generator matrix A , and how to handle the case where the transition matrix P undergoes a rank-one change. Furthermore, the relationship between this method and Gaussian elimination is exploited to develop a more stable, GTH-like variant that can handle nearly uncoupled systems and other situations where the original algorithm fails.

2. The algorithm

The direct projection method can be derived in several different ways. Here the method is introduced through a geometric construction, as in [2,3]. This process is referred to as the *null vector algorithm*. Let a_i^T denote the i th row of A , and let $\mathcal{N}_k := \mathcal{N}(A_k)$ denote the null space of the $k \times n$ matrix

$$A_k = \begin{bmatrix} a_1^T \\ a_2^T \\ \vdots \\ a_k^T \end{bmatrix}.$$

Then

$$\mathcal{N}_1 \supset \mathcal{N}_2 \supset \cdots \supset \mathcal{N}_{n-1} = \mathcal{N}_n = \text{span}\{\pi^T\}.$$

The idea of the method is to construct a sequence of sets

$$\mathcal{B}_k = \{z_{k+1}^{(k)}, z_{k+2}^{(k)}, \dots, z_n^{(k)}\}, \quad k = 0, 1, \dots, n-1,$$

such that \mathcal{B}_k is a basis for \mathcal{N}_k . For $k = 0$, one can take

$$\mathcal{B}_0 = \{z_1^{(0)}, z_2^{(0)}, \dots, z_n^{(0)}\}$$

to be any basis for \mathbb{R}^n . Each basis set \mathcal{B}_k is then obtained from \mathcal{B}_{k-1} by projecting the vectors $z_{k+1}^{(k-1)}, z_{k+2}^{(k-1)}, \dots, z_n^{(k-1)} \in \mathcal{N}_{k-1}$ onto a_k^\perp along the direction of $z_k^{(k-1)}$:

$$z_j^{(k)} = \left(I_n - \frac{z_k^{(k-1)} a_k^T}{a_k^T z_k^{(k-1)}} \right) z_j^{(k-1)}, \quad j = k+1, \dots, n.$$

It is immediate to verify that $z_j^{(k)} \in \mathcal{N}_k$ for $j = k+1, \dots, n$.

In order for the algorithm to be well defined, it is necessary that $a_k^T z_k^{(k-1)} \neq 0$, that is, $z_k^{(k-1)} \notin \mathcal{N}_k$. Provided that $\text{rank}(A_k) = k$, this can always be ensured, if necessary, by renumbering the vectors $z_k^{(k-1)}, \dots, z_n^{(k-1)}$; and indeed, for numerical stability it was recommended in [3] to choose the direction vector $z_k^{(k-1)}$ so that

$$\left| a_k^T z_k^{(k-1)} \right| = \max_{k \leq j \leq n} \left| a_k^T z_j^{(k-1)} \right|. \tag{2}$$

Note that the pivoting rule (2) admits the following geometric interpretation: among all the vectors $z_k^{(k-1)}, z_{k+1}^{(k-1)}, \dots, z_n^{(k-1)}$, choose as the projection direction the one that has maximum distance to the hyperplane a_k^\perp . Also note that if the vectors have been normalized in the Euclidean norm, rule (2) prescribes to pick the projection direction which is the most nearly perpendicular to the hyperplane a_k^\perp .

It is easy to show that as long as $a_k^T z_k^{(k-1)} \neq 0$, each of the sets $\mathcal{B}_k = \{z_{k+1}^{(k)}, \dots, z_n^{(k)}\}$ thus constructed is indeed a basis for \mathcal{N}_k , for $k = 1, 2, \dots, n-1$;

see [2,3]. In particular, $\mathcal{B}_{n-1} = \{z_n^{(n-1)}\}$ where $z_n^{(n-1)}$ is a nonzero multiple of the stationary vector $x = \pi^T$.

Assuming that $a_k^T z_k^{(k-1)} \neq 0$ for all $k = 1, 2, \dots, n-1$, no pivoting is necessary and the null vector algorithm may be formalized as follows:

Algorithm 1. Null Vector Algorithm

Let $\{z_1^{(0)}, z_2^{(0)}, \dots, z_n^{(0)}\}$ be a basis of \mathbb{R}^n .

for $k = 1 : n - 1$

for $j = k : n$

$$d_j^{(k-1)} := a_k^T z_j^{(k-1)}$$

end

for $j = k + 1 : n$

$$z_j^{(k)} := z_j^{(k-1)} - \left(\frac{d_j^{(k-1)}}{d_k^{(k-1)}} \right) z_k^{(k-1)} \quad (3)$$

end

end

At step k of the algorithm, it may be desirable to normalize the vector $z_{k+1}^{(k)}$, for example in the ℓ_1 -norm, immediately after it is computed in (3). Incidentally, it should be noted that a_n^T , the last row of A , is redundant and is not needed in Algorithm 1. Although in this paper symmetric ($A = A^T$) problems are not of interest, it is worth pointing out that when A is symmetric and positive definite, Algorithm 1 amounts to Gram–Schmidt orthogonalization of the basis vectors $z_1^{(0)}, z_2^{(0)}, \dots, z_n^{(0)}$ with respect to the inner product $\langle x, y \rangle_A = x^T A y$. In other words, Algorithm 1 is a conjugate direction method. In this form, the algorithm was proposed by Fox et al. in [16] and therefore it predates Purcell’s contribution by a few years; see also [23, p. 412]. In the interest of historical accuracy, it should also be noted that this algorithm is equivalent to an even earlier method, the *escalator process* of Morris; see [33] and Faddeev and Faddeeva [13, pp. 168–173 and 391–392]. See also [38].

The null vector algorithm is actually a whole family of algorithms, since there are infinitely many possible choices of the initial basis set \mathcal{B}_0 . A natural and computationally convenient choice of the initial basis \mathcal{B}_0 is given by the standard basis $\{e_1, e_2, \dots, e_n\}$; that is, $z_i^{(0)} = e_i$ for all $i = 1, 2, \dots, n$. In this case, the resulting direct projection method is intimately related to Gaussian elimination; it is sometimes referred to as *implicit LU decomposition*. To demonstrate this relationship, let $Z^{(k)}$ be the $n \times (n - k)$ matrix defined by

$$Z^{(k)} = \begin{bmatrix} z_{k+1}^{(k)} & z_{k+2}^{(k)} & \dots & z_n^{(k)} \end{bmatrix}, \quad k = 0, 1, \dots, n - 1.$$

Note that $Z^{(0)} = I_n$. Assuming $d_1^{(0)} := a_1^T z_1^{(0)} = a_1^T e_1 = a_{11} \neq 0$, one can project e_2, e_3, \dots, e_n along e_1 onto $\mathcal{N}_1 = a_1^\perp$ to obtain the set $\mathcal{B}_1 = \{z_2^{(1)}, z_3^{(1)}, \dots, z_n^{(1)}\}$ where

$$z_j^{(1)} = e_j - \left(\frac{a_{1j}}{a_{11}}\right) e_1, \quad j = 2, 3, \dots, n.$$

Hence, $Z^{(1)}$ is the $n \times (n - 1)$ matrix

$$Z^{(1)} = \begin{bmatrix} -\frac{a_{12}}{a_{11}} & -\frac{a_{13}}{a_{11}} & \dots & -\frac{a_{1n}}{a_{11}} \\ 1 & 0 & \dots & 0 \\ 0 & 1 & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & 1 \end{bmatrix} = [z_2^{(1)}, z_3^{(1)}, \dots, z_n^{(1)}].$$

It is obvious that the columns of $Z^{(1)}$ are linearly independent. Now, it will be possible to project $z_3^{(1)}, \dots, z_n^{(1)}$ onto a_2^\perp along $z_2^{(1)}$ to obtain a basis for \mathcal{N}_2 if and only if the condition $a_2^T z_2^{(1)} \neq 0$ is satisfied. Computing $d_2^{(1)} := a_2^T z_2^{(1)}$ one finds

$$d_2^{(1)} = a_{22} - \frac{a_{21}a_{12}}{a_{11}} = \frac{\begin{vmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{vmatrix}}{a_{11}}.$$

Hence, provided that the leading 2×2 principal minor of A is nonzero, the next round of projections can be performed to compute $\mathcal{B}_2 = \{z_3^{(2)}, \dots, z_n^{(2)}\}$. It is clear that the columns of $Z^{(2)} = [z_3^{(2)}, \dots, z_n^{(2)}]$ are linearly independent, since the $n \times (n - 2)$ matrix $Z^{(2)}$ has the form

$$Z^{(2)} = \begin{bmatrix} * & * & \dots & * \\ * & * & \dots & * \\ 1 & 0 & \dots & 0 \\ 0 & 1 & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & 1 \end{bmatrix}.$$

Therefore \mathcal{B}_2 is a basis of \mathcal{N}_2 . A simple inductive argument shows that the algorithm can be carried to completion if all the leading principal minors of order k , for $k = 1, 2, \dots, n - 1$, are nonzero. It is well known that such property holds for singular irreducible M -matrices (see [7, p. 156]) and in particular for the generator matrix of a finite ergodic Markov chain. Unless otherwise specified, in the remainder of the paper A is assumed to be such a matrix, but many of the ensuing statements remain true under more general assumptions. For example, it is enough to assume that all states in the underlying Markov chain lead to state n . This means that for each $i = 1, 2, \dots, n - 1$, there is a sequence of distinct states $j = i_1, i_2, \dots, i_k = n$ with

$$p_{i_1 i_2} > 0, p_{i_2 i_3} > 0, \dots, p_{i_{k-1} i_k} > 0;$$

see the discussion in [35, pp. 507–508].

The resulting sequence of matrices $Z^{(k)} \in \mathbb{R}^{n-k}$, $k = 1, 2, \dots, n-1$, satisfies (by construction) $A_k Z^{(k)} = O$; in particular, the column vector $Z^{(n-1)} = [z_n^{(n-1)}]$ spans the null space of A_{n-1} (which coincides with the null space of A). It is also straightforward that all the column vectors $z_i^{(k)}$ have nonnegative entries, that all the entries of $z_i^{(k)}$ below the i th one are zero, and that the k th entry of $z_k^{(k-1)}$ is equal to 1, unless normalization is used in the course of the null vector algorithm. In particular, the last entry of $z_n^{(n-1)}$ is equal to 1.

Hence, the $n \times n$ matrix of null vectors

$$Z := [z_1^{(0)}, z_2^{(1)}, \dots, z_n^{(n-1)}]$$

is a unit upper triangular matrix with nonnegative entries with the property that the (i, j) entry of AZ is zero for $j > i$ (this follows immediately from the fact that columns $k+1$ through n of Z form a basis for \mathcal{N}_k). In symbols:

$$AZ = A, \quad A \text{ lower triangular.}$$

Now, it is well known that a singular, irreducible M -matrix A can be decomposed uniquely as $A = LDU$ where L and U are unit lower and upper triangular matrices (respectively) and D is a diagonal matrix of rank $n-1$:

$$D = \text{diag}(d_1, d_2, \dots, d_{n-1}, 0), \quad d_i > 0 \quad \text{for } 1 \leq i \leq n-1;$$

see, e.g., [41]. It follows necessarily that $Z = U^{-1}$ and that $A = LD$. In particular, the entries on the main diagonal of A coincide with those of D :

$$d_i = d_i^{(i-1)} = a_i^T z_i^{(i-1)} = A_{ii}, \quad i = 1, 2, \dots, n.$$

Thus the d_i 's coincide with the pivots arising in the Gaussian elimination process. Note that $d_n = 0$, expressing the fact that $z_n^{(n-1)}$, the last column of Z , is a null vector of A (and therefore a multiple of the stationary distribution vector $x = \pi^T$ of the Markov chain).

Hence, the null vector Algorithm 1 with the initial choice $Z^{(0)} = I_n$ is closely related to Gaussian elimination: it is an *implicit LDU* factorization in the sense that it produces the inverse factor U^{-1} and the diagonal matrix of pivots D , but not the matrix L . Actually, as shown already in [23, pp. 426–427], the algorithm also computes the entries of U , which are given (for $j > k$) by $u_{kj} = d_j^{(k-1)} / d_k^{(k-1)}$, i.e., by the multipliers in the updates (3). Thus the algorithm simultaneously computes U and U^{-1} ; the first row-wise, the second column-wise. Since the entries of U are not needed, they can be discarded.

In [2,3] it is shown how the factor Z , together with the lower triangular part of A , is all one needs to solve nonsingular systems $Ax = b$ for arbitrary right-hand sides b in $O(n^2)$ arithmetic operations. This solve phase is again a direct projection process, whereby an arbitrary initial vector is successively projected onto

the hyperplanes defined by the individual equations $a_i^T x = b_i$ along the projection directions $z_i^{(i-1)}$, leading to the unique solution $x^* = A^{-1}b$ after n such projections. In the case of a singular, homogeneous system $Ax = 0$ this solve phase is not needed, since the solution is just the (normalized) last column of Z (but see Section 4).

It is natural to compare the direct projection method to Gaussian elimination (more precisely, LDU factorization) for the solution of Markov chains. In Gaussian elimination, the factorization $A = LDU$ is first computed. Unless the L factor is needed for other purposes, it can be discarded. The system $Ax = 0$ is then equivalent to $DUx = 0$, or $Ux = \alpha e_n$ since $D = \text{diag}(d_1, d_2, \dots, d_{n-1}, 0)$. Setting the scaling factor $\alpha = 1$, this system has the unique solution $x = U^{-1}e_n$; note that this is, again, the last column of $Z = U^{-1}$. Finally, the vector x is normalized to yield the stationary vector π^T .

Hence, Gaussian elimination requires two phases, factorization and backsubstitution. In contrast, no backsubstitution is necessary in the direct projection method. In this regard, there is a resemblance between the direct projection method and QR factorization. Indeed, the last column in the Q factor of $A^T = I_n - P$ yields a multiple of the stationary distribution vector. Although significantly more expensive, the QR factorization can be used to gauge the sensitivity of the stationary distribution to perturbations in the transition probabilities; see [21].

What about costs? Computing the stationary vector by Gaussian elimination followed by backsubstitution requires $\frac{2}{3}n^3 + O(n^2)$ arithmetic operations, and it is easy to see that the null vector algorithm with initial basis $Z^{(0)} = I_n$ has exactly the same arithmetic complexity. Note that when implementing the algorithm, the structure of the vectors $z_i^{(k)}$ must be taken into account so as to avoid operations with zeros, but this is very easy to do.

A possible advantage of the direct projection method, already mentioned in Purcell's original paper [37], is that it can be implemented using only $\frac{1}{4}n^2 + O(n)$ memory locations. This is possible because each row a_k^T is needed only at the k th step in Algorithm 1. Hence, the rows of A can be generated one at a time and then discarded after they have been used to update the $z_i^{(k)}$ vectors in the null vector algorithm. Moreover, at step k of the algorithm only the null vectors $z_{k+1}^{(k)}, \dots, z_n^{(k)}$ need to be kept in storage. Exploiting the zero structure of these vectors leads to an algorithm requiring only about one fourth of the entries required by standard implementations of Gaussian elimination. It is mentioned in [41, pp. 82–83] that Gaussian elimination can be efficiently implemented even when the generator A (or A^T) is generated row-by-row. However, all $\frac{1}{2}n^2$ entries of U are needed to perform the backsubstitution phase. Hence, even in this case, the direct projection method still requires only about half the storage.

Finally, it has been shown in [9] that when pivoting is not needed (as is generally the case for Markov chain problems), efficient parallel implementations of the direct projection algorithm are possible.

3. Computation of generalized inverses

Let $A = LDU$ be the LDU factorization $A = LDU$, where L and U are unit lower and upper triangular matrices (respectively) and D is a diagonal matrix of rank $n - 1$:

$$D = \text{diag}(d_1, d_2, \dots, d_{n-1}, 0), \quad d_i > 0 \quad \text{for } 1 \leq i \leq n - 1.$$

Notice that L and U are nonsingular M -matrices; in particular, L^{-1} and U^{-1} have nonnegative entries. Let

$$A^- = U^{-1}D^-L^{-1}, \quad \text{where } D^- = \text{diag}(d_1^{-1}, d_2^{-1}, \dots, d_{n-1}^{-1}, 0).$$

Proposition 1. *The matrix A^- satisfies the first two of Penrose's equations [8], namely:*

$$AA^-A = A \quad \text{and} \quad A^-AA^- = A^-. \quad (4)$$

Proof. It suffices to observe that

$$AA^-A = (LDU)(U^{-1}D^-L^{-1})(LDU) = LDD^-DU$$

and that

$$A^-AA^- = (U^{-1}D^-L^{-1})(LDU)(U^{-1}D^-L^{-1}) = U^{-1}D^-DD^-L^{-1}.$$

The result then follows from the obvious identities

$$DD^-D = D \quad \text{and} \quad D^-DD^- = D^-. \quad \square$$

The first of the two identities (4) states that A^- is an *inner inverse* of A and the second that A^- is an *outer inverse* of A . A generalized inverse satisfying these two conditions is called a $\{1, 2\}$ -inverse of A or an *inner–outer inverse*. (Another term that is found in the literature is *reflexive inverse*; see [8, p. 96].) Because A^- does not necessarily satisfy the third and fourth Penrose conditions, it is not the Moore–Penrose pseudoinverse A^\dagger of A in general. Because A^\dagger is obviously a $\{1, 2\}$ -inverse, this kind of generalized inverse is nonunique. Indeed, there are infinitely many such $\{1, 2\}$ -inverses in general. Each pair R, N of subspaces of \mathbb{R}^n that are complements of the null space and range of A (respectively) uniquely determines a $\{1, 2\}$ -inverse $G_{N,R}$ of A with null space $\mathcal{N}(G_{N,R}) = N$ and range $\mathcal{R}(G_{N,R}) = R$; see [8]. In the case of A^- it is readily verified that $N = \text{span}\{e_n\}$ and $R = \text{span}\{e_1, e_2, \dots, e_{n-1}\}$ where e_i denotes the i th unit basis vector in \mathbb{R}^n . It is easy to see that R is complementary to $\mathcal{N}(A)$ and N is complementary to $\mathcal{R}(A)$. The pseudoinverse A^\dagger corresponds to $R = \mathcal{R}(A^T)$, $N = \mathcal{N}(A^T)$.

An obvious way to compute A^- is to first compute the LDU factorization of A , and then to invert L and U (and the nonzero entries in D). This requires $\frac{2}{3}n^3 + O(n^2)$ for the factorization, and $\frac{1}{3}n^3 + O(n^2)$ for each of the triangular matrix inversions, for a total of $\frac{4}{3}n^3 + O(n^2)$ operations. Alternatively, Algorithm 1 can be used to

compute A^- by first running the null vector algorithm on A to obtain Z and D , and then on A^T to obtain a unit upper triangular matrix W which is immediately seen to coincide with L^{-T} (the entries of D need not be recomputed). While the two procedures require almost exactly the same number of arithmetic operations, the one based on Algorithm 1 would be advantageous on a shared memory computer with at least two processors, since the computation of U^{-1} and that of L^{-1} can be carried out completely independently of one another. In contrast, the computation of L and that of U in Gaussian elimination are inextricably connected. Even if the subsequent inversion of L and U is carried out in parallel on two processors, the procedure based on Algorithm 1 can be expected to be faster by a factor of one-third.

It is straightforward to check that A^-A is the oblique projector onto $R = \text{span}\{e_1, e_2, \dots, e_{n-1}\}$ along $\mathcal{N}(A)$ and that AA^- is the oblique projector onto $\mathcal{R}(A)$ along $N = \text{span}\{e_n\}$. Therefore A^-A has eigenvalues 0 with multiplicity 1, and 1 with multiplicity $n - 1$; likewise for AA^- . This observation suggests that inexpensive approximations $M \approx A^-$ can be used as preconditioners for Krylov subspace methods applied to large-scale problems (for which direct methods may be too expensive), since in this case most eigenvalues of the preconditioned matrix MA will be clustered around 1. Sparse approximations to Z (and W) can be obtained by performing numerical dropping after each update (3) in Algorithm 1. For example, entries in the newly computed vectors $z_j^{(k)}$ can be dropped if their magnitude falls below a preset drop tolerance $0 < \tau < 1$. This approach has been developed, with excellent results, in [4] for the positive definite case, in [5] for general sparse matrices, and in [6] for Markov chain problems. One notable advantage of this type of preconditioning over incomplete LU factorization methods is that the application of the preconditioner can be readily parallelized, since it consists of matrix-vector multiplies, rather than triangular solves.

It is well known that in the Markov chain context, the most relevant generalized inverse is $A^\#$, the *group inverse* (or *Drazin inverse*); this is the unique matrix that satisfies (4) and the additional condition $AA^\# = A^\#A$. See [8, Chapter 8; 29]. Virtually all the quantities of interest in the analysis of a finite Markov chain can be easily obtained from this matrix. In general, the $\{1, 2\}$ -inverse A^- is different from $A^\#$. This can be seen from the fact that in general $AA^- \neq A^-A$. Also notice that for a singular irreducible M -matrix A the $\{1, 2\}$ -inverse $A^- = U^{-1}D^-L^{-1}$ is always a nonnegative matrix, which is not true in general for either the group or the Moore–Penrose inverse. As the following straightforward result shows, however, there is a close relationship between A^- and $A^\#$.

Proposition 2. *Let $A = I_n - P^T$ be the generator matrix of an ergodic Markov chain, and let $x = \pi^T$ be the corresponding stationary distribution vector. Then the group inverse of A is given by*

$$A^\# = (I_n - xe^T)A^-(I_n - xe^T), \quad (5)$$

where $e \in \mathbb{R}^n$ is the vector of all ones.

Proof. Using the well known fact (see, e.g., [18]) that the group inverse satisfies $AA^\# = A^\#A = I_n - xe^T$, it follows that

$$\begin{aligned} (I_n - xe^T)A^-(I_n - xe^T) &= (A^\#A)A^-(AA^\#) \\ &= A^\#(AA^-A)A^\# = A^\#AA^\# = A^\#. \quad \square \end{aligned}$$

Obviously, the result remains true if any inner inverse is used instead of A^- .

Once the $\{1, 2\}$ -inverse $A^- = U^{-1}D^-L^{-1} = ZD^-W^T$ has been computed, and with it the stationary vector $\pi^T = z_n^{(n-1)} / \|z_n^{(n-1)}\|_1$, the group inverse $A^\#$ can be easily computed at a cost of an additional $O(n^2)$ arithmetic operations on the basis of (5).

The number of arithmetic operations required by the algorithm just described is $\frac{4}{3}n^3 + O(n^2)$, or half as many as the standard algorithm based on the LU factorization of A (see [18; 41, p. 119]). If, moreover, advantage is taken of the fact that the inverse factors Z and W can be computed concurrently with no communication involved, the time to compute $A^\#$ via Algorithm 1 and Theorem 2 is proportional to just about $\frac{2}{3}n^3 + O(n^2)$. Hence, $A^\#$ can be computed in essentially the same time needed to get π .

It is frequently observed (for example, in [24]) that unless the individual entries of $A^\#$ are of interest, there is no need to compute $A^\#$ explicitly. Indeed all the quantities of interest, like for instance the means and variances of the first passage/return times, can be computed just using the triangular factors from Gaussian elimination (or from the GTH algorithm) and the stationary vector π . When the GTH algorithm is used rather than an explicitly computed $A^\#$, one has the added benefit of (componentwise) more accurate solutions [24]. The algorithm that was found in [24] to be the most effective was one involving the explicit inversion of one of the triangular factors computed by the GTH algorithm. Since the algorithm studied in this paper produces such an inverse factor directly, it should be useful for computing a number of quantities of interest in Markov chain analysis, and not just the stationary vector.

4. Updating the stationary vector

In Markov chain modeling one is frequently interested in knowing how the stationary vector π changes when the probabilities of leaving one of the states (say, state i) for the remaining states are changed. This leads to the problem of how to efficiently update the stationary vector when the transition probability matrix P undergoes a rank-one change (row modification) of the form

$$\tilde{P} = P + e_i c^T, \quad c \in \mathbb{R}^n, \quad 0 \leq p_{ij} + c_j < 1, \quad c^T e = 0.$$

Here “efficiently” means in $O(n^2)$ arithmetic operations, as opposed to the $O(n^3)$ that would be necessary if the solution were to be computed from scratch. Similarly, it is sometimes important to update the group inverse in $O(n^2)$ operations. These

problems have been studied by several authors; see, e.g., [20,26,28,32]. In particular, in [20] it is shown how to update the LU factors of $\tilde{A} = I_n - \tilde{P}^T = A - ce_i^T$ at the cost of $O(n^2)$ operations.

The direct projection method is also able to handle such a situation without difficulties. It is also possible to update the group inverse $\tilde{A}^\#$ in $O(n^2)$ operations. It is assumed here that the change from P to \tilde{P} is “ergodic-to-ergodic”; that is, irreducibility is preserved.

It is easy to verify (see, e.g., [20, p. 31]) that if $x \in \mathbb{R}^n$ is a solution to the homogeneous system $Ax = 0$ and $y \in \mathbb{R}^n$ is a solution to the nonhomogeneous system $Ay = c$, then

$$\tilde{x} = x - \left(\frac{e_i^T x}{e_i^T y - 1} \right) y \quad \text{solves } \tilde{A}\tilde{x} = 0. \tag{6}$$

Note that the condition $c^T e = 0$ guarantees that the linear system $Ay = c$ is consistent, since $c \in e^\perp = \mathcal{N}(A^T)^\perp = \mathcal{R}(A)$. Also, the assumption that irreducibility is preserved ($\text{rank}(\tilde{A}) = \text{rank}(A) = n - 1$) ensures, by a result of Wedderburn, that $e_i^T y \neq 1$; see [43, p. 69]. Upon normalization, \tilde{x} yields the stationary distribution vector for the modified matrix \tilde{P} .

If A^- is any inner inverse of A , and $b \in \mathcal{R}(A)$, then A^-b is a solution of $Ax = b$; see [8]. It follows that if $A^- = ZD^-W^T$ is available, the stationary vector $\tilde{x} = \tilde{\pi}^T$ of the modified matrix \tilde{P} can be computed in $O(n^2)$ operations as

$$\tilde{x} = x - \left(\frac{e_i^T x}{e_i^T A^-c - 1} \right) A^-c.$$

If the W factor is not explicitly available, an alternative way to proceed is to compute a solution of the nonhomogeneous system $Ay = c$ using the projection directions z_2, z_3, \dots, z_{n-1} , with $z_k := z_k^{(k-1)}$. Let

$$\mathcal{H}_k = \{x \in \mathbb{R}^n \mid a_k^T x = c_k\}$$

be the hyperplane corresponding to the k th equation in the system $Ay = c$. Note that the solution set of this system of equations is given by

$$\mathcal{S} = \bigcap_{k=1}^n \mathcal{H}_k = \bigcap_{k=1}^{n-1} \mathcal{H}_k,$$

where the second identity is a consequence of the fact that the last equation in the system is necessarily a linear combination of the first $n - 1$ equations. Now let $x^{(1)} \in \mathbb{R}^n$ be any point in \mathcal{H}_1 , i.e., any solution to the first equation in the system. For example, $x^{(1)} = (c_1/a_{11})e_1$. A solution to $Ay = c$ can be obtained by a sequence of $n - 2$ projections, as follows:

1. Project $x^{(1)}$ onto \mathcal{H}_2 along z_2 to obtain

$$x^{(2)} = x^{(1)} + \frac{c_2 - a_2^T x^{(1)}}{a_2^T z_2} z_2 = x^{(1)} + \frac{c_2 - a_2^T x^{(1)}}{d_2} z_2 \in \mathcal{H}_1 \cap \mathcal{H}_2.$$

2. Project $x^{(2)}$ onto \mathcal{H}_3 along z_3 to obtain

$$x^{(3)} = x^{(2)} + \frac{c_3 - a_3^T x^{(2)}}{a_3^T z_3} z_3 = x^{(2)} + \frac{c_3 - a_3^T x^{(2)}}{d_3} z_3 \in \mathcal{H}_1 \cap \mathcal{H}_2 \cap \mathcal{H}_3.$$

⋮

$n - 2$. Project $x^{(n-2)} \in \cap_{k=1}^{n-2} \mathcal{H}_k$ onto \mathcal{H}_{n-1} along z_{n-1} to obtain

$$x^{(n-1)} = x^{(n-2)} + \frac{c_{n-1} - a_{n-1}^T x^{(n-2)}}{d_{n-1}} z_{n-1} \in \bigcap_{k=1}^{n-1} \mathcal{H}_k = \mathcal{S}.$$

Hence $y := x^{(n-1)}$ is a particular solution of $Ay = c$. Note that in order to carry out this solution process it is necessary to have the projection directions $z_1 = e_1, z_2, \dots, z_{n-1}$, the pivots $d_1 = a_{11}, d_2, \dots, d_{n-1}$, and the entries a_{ij} of A with $i = 2, \dots, n-1$ and $j < i$. The overall cost is $O(n^2)$ arithmetic and storage. The resulting $y = x^{(n)}$ is then used to update the stationary vector \tilde{x} according to (6). The advantage of this procedure over the one that uses $y = ZD^{-1}W^T c$ is that it is not necessary to compute the triangular W factor explicitly.

As long as only one column of A is modified (possibly several times in sequence), the strategies outlined above work just fine. A problem arises when a modification to column i of A is followed by a modification to column $k \neq i$, since in this case it is required to solve a homogeneous system of the form $(A - ce_k^T)y = v$, where $e^T v = 0$; see [20]. Unfortunately, this cannot be done with Z and $A - ce_k^T$ alone. It is better in this case to compute the updated factor \tilde{Z} of $\tilde{A} = A - ce_k^T$; then the last column of \tilde{Z} directly yields the updated stationary vector \tilde{x} .

For the nonsingular case, the problem of updating Z has been investigated by Fletcher [14], who showed how this can be done in $O(n^2)$ operations. Interestingly, Fletcher was able to show that updating $Z = U^{-1}$ can be done more efficiently than updating U itself. (Actually the analysis in [14] is done for $W^T = L^{-1}$, but the same holds for $Z = U^{-1}$.) The same update procedure can be applied in the Markov chain context.

Similarly, it is possible to update the group inverse in $O(n^2)$ operations as follows. First, the updated stationary distribution vector \tilde{x} is computed. Then, a $\{1, 2\}$ -inverse of $\tilde{A} = A - ce_i^T$ is computed in $O(n^2)$ work from A^- via the following generalization of the Sherman–Morrison formula:

$$(A - ce_i^T)^- = A^- - \frac{A^- ce_i^T A^-}{e_i^T A^- c - 1}$$

(see [28, Theorem 7; third case since $c \in \mathcal{R}(A)$]). Finally, the updated group inverse $\tilde{A}^\#$ of $\tilde{A} = A - ce_i^T$ is given by

$$\tilde{A}^\# = (I_n - \tilde{x}e^T)(A - ce_i^T)^-(I_n - \tilde{x}e^T).$$

Since each step in this process costs $O(n^2)$ operations, the total cost of updating the group inverse is also $O(n^2)$.

5. A more stable variant

When applied to irreducible weakly diagonally dominant matrices of the form $A = I_n - P^T$, Gaussian elimination is backward stable; see [17,19]. On the other hand, it is well known that there are important examples of Markov chains for which Gaussian elimination fails to compute an accurate stationary vector. The following example is taken from [40]. Consider the transition probability matrix

$$P = \begin{bmatrix} \frac{1}{2} - \epsilon & \frac{1}{2} & \epsilon \\ \frac{1}{2} & \frac{1}{2} - \epsilon & \epsilon \\ \epsilon & \epsilon & 1 - 2\epsilon \end{bmatrix}.$$

If ϵ is less than machine precision, then the generator matrix is rounded to

$$\bar{A} = \text{fl}(I_3 - P^T) = \begin{bmatrix} \frac{1}{2} & -\frac{1}{2} & -\epsilon \\ -\frac{1}{2} & \frac{1}{2} & -\epsilon \\ -\epsilon & -\epsilon & 0 \end{bmatrix}.$$

When applied to this matrix, Gaussian elimination breaks down owing to a zero pivot in the second step. However, Gaussian elimination can hardly be blamed: indeed, the act of explicitly forming the generator matrix A has changed the nature of the problem, and Gaussian elimination (or any other backward stable algorithm, for example QR) cannot be expected to deliver an accurate answer. In the words of [39], Gaussian elimination is being asked to do the impossible: solving a problem that is not in the computer. It is also easy to construct examples where no digits are lost when forming $A = I_n - P^T$ and no zero pivot occurs, but the computed solution is still affected by large errors. Being backward stable, Gaussian elimination will produce the exact solution to a perturbed problem of the form $(A + E)x = 0$, where the entries of E are bounded by a small multiple of machine epsilon. (Note that the entries of the generator matrix A are bounded by 1 in magnitude.) Unfortunately, on examples like the one above such a perturbation cannot be considered small. In this sense, the matrix $A = I_n - P^T$ is very ill-conditioned. Similar difficulties occur when P is a small perturbation of the identity; see [12].

The *coupling matrices* that arise when aggregation–disaggregation techniques [41] are applied to nearly uncoupled chains are small perturbations of the identity. A Markov chain is *nearly uncoupled* when the transition matrix P can be partitioned as

$$P = \begin{bmatrix} P_{11} & E_{12} & \dots & E_{1N} \\ E_{21} & P_{22} & \dots & E_{2N} \\ \vdots & \vdots & \ddots & \vdots \\ E_{N1} & E_{N2} & \dots & P_{NN} \end{bmatrix},$$

where the off-diagonal blocks E_{ij} have small norm. Writing

$$P = \text{diag}(P_{11}, P_{22}, \dots, P_{NN}) + E,$$

the *degree of coupling* of the chain is defined as $\gamma = \|E\|_\infty$, and is a measure of the decomposability of the matrix. For $\gamma = 0$, the matrix is decomposable (reducible). When γ is small, the matrix is nearly uncoupled and the corresponding coupling matrix is a small perturbation (of the order of γ) of the $N \times N$ identity.

The sensitivity of the stationary vector of a Markov chain with respect to perturbations in the transition probabilities has been studied by several authors; see, e.g., [1,27,31,34,39,42,44]. In particular, O’Cinneide has shown in [34] that when the data are taken to be the transition probabilities p_{ij} with $i \neq j$, which completely determine the chain, the problem of computing the stationary vector π is well-conditioned in the sense of entrywise relative error, regardless of the degree of coupling of the chain. Therefore, in principle, it should be possible to compute the probabilities π_i to high relative accuracy.

Grassmann et al. [22] have proposed a modification of Gaussian elimination that gives accurate solutions for any ergodic chain. The GTH method uses the probabilities p_{ij} ($i \neq j$) as the data. In the first step, a_{11} is computed as $\sum_{j=2}^n p_{j1}$ rather than as $1 - p_{11}$. At step $k + 1$, the algorithm uses the fact that the active $(n - k) \times (n - k)$ submatrix in Gaussian elimination is also a singular irreducible M -matrix with zero column sums; indeed, it is the Schur complement in A of the leading $k \times k$ principal submatrix of A (see Lemma 3). Thus, the current pivot is given by the negated sum of the off-diagonal elements in the corresponding (unreduced) column. It is easy to check that this algorithm gives the correct answer for the example above. Actually, the original GTH algorithm works with $A^T = I_n - P$ rather than A , and it computes a lower triangular matrix L and a unit upper triangular matrix U such that $A^T = UL$. A backsubstitution involving U is then used to compute the stationary vector. Different implementations of this algorithm are possible and can be found in [41].

The authors of [22] suggested that the algorithm’s accuracy stems from the fact that it involves no subtractions, and all the quantities involved are nonnegative. Therefore, loss of accuracy due to cancellation cannot occur. A formal error analysis by O’Cinneide [34] supports this intuition. The main result of [34] is that the relative error in the entries of the stationary vector computed by GTH is bounded by roughly $9n^2\mathbf{u}$, where \mathbf{u} denotes the unit round-off; in practice the actual error is usually much smaller. This result is valid independently of any structure the matrix P may have, and it applies to nearly uncoupled chains as well as to any other type of chain. The GTH algorithm is now the method of choice in many applications, and is routinely used to compute the stationary vector of the coupling matrices arising in aggregation–disaggregation methods; see, for instance, [12].

The situation for the direct projection algorithm closely parallels that for Gaussian elimination. Fletcher’s round-off error analysis [14] shows that the computation of $Z = U^{-1}$ is backward stable. More precisely, there exists an upper triangular

matrix E such that the computed Z and D factors are exact factors of $A + E$, where the entries of E are bounded by $(\frac{3}{2}n^2 + O(n))\mathbf{u}\rho + O(\mathbf{u}^2)$. Here ρ is an appropriate growth factor for the entries of Z , which behaves similarly to the growth factor in Gaussian elimination. The error analysis in [14] was carried out for the case of nonsingular A , but the same result holds in the present context.

The algorithm fails on the same problems that cause trouble for Gaussian elimination. The question therefore arises of whether it is possible to develop a GTH-like version of Algorithm 1 that can be expected to compute the stationary probabilities accurately. The answer is affirmative, and the remainder of this section is devoted to developing such a variant.

The following result from [30] will be needed:

Lemma 3. *Let P be an irreducible row-stochastic matrix partitioned as*

$$\begin{bmatrix} P_{11} & P_{12} \\ P_{21} & P_{22} \end{bmatrix}.$$

Assume that

$$A = \begin{bmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{bmatrix} = \begin{bmatrix} I_k - P_{11}^T & -P_{21}^T \\ -P_{12}^T & I_{n-k} - P_{22}^T \end{bmatrix}.$$

Then $S = A_{22} - A_{21}A_{11}^{-1}A_{12}$, the Schur complement of A_{11} in A , is a singular irreducible M -matrix with a one-dimensional null space.

The Schur complement $S = A_{22} - A_{21}A_{11}^{-1}A_{12}$ has the following probabilistic interpretation: it is the generator matrix for the chain observed only while in the states $\{k + 1, k + 2, \dots, n\}$. As is well known, S coincides with the active $(n - k) \times (n - k)$ submatrix obtained after k steps of Gaussian elimination applied to $Ax = 0$.

Lemma 4. *Let P and A be as in Lemma 3 and let $Z^{(k)}$ be the $n \times (n - k)$ matrix of null vectors obtained after k steps of Algorithm 1, where $1 \leq k \leq n - 1$. Then*

$$Z^{(k)} = \begin{bmatrix} -A_{11}^{-1}A_{12} \\ I_{n-k} \end{bmatrix}.$$

Proof. Write

$$A_k = [A_{11} \quad A_{12}] \quad \text{and} \quad Z^{(k)} = \begin{bmatrix} Z_k \\ I_{n-k} \end{bmatrix}.$$

Since the columns of $Z^{(k)}$ span $\mathcal{N}_k = \mathcal{N}(A_k)$, it is $A_k Z^{(k)} = O$ (the null matrix of order k), or

$$A_{11}Z_k + A_{12} = O.$$

But A_{11} is nonsingular, therefore $Z_k = -A_{11}^{-1}A_{12}$. \square

The following result is the basis for the GTH-like variant of Algorithm 1.

Theorem 5. *The pivot at step $k + 1$ in Algorithm 1 (with $Z^{(0)} = I_n$) is given by*

$$d_{k+1}^{(k)} = \left(- \sum_{i=k+2}^n a_i^T \right) z_{k+1}^{(k)}. \quad (7)$$

Proof. By Lemma 4 the Schur complement $S = A_{22} - A_{21}A_{11}^{-1}A_{12}$ is given by

$$S = \begin{bmatrix} A_{21} & A_{22} \end{bmatrix} Z^{(k)}.$$

The pivots for the null vector algorithm initialized with $Z^{(0)} = I_n$ coincide with the pivots in Gaussian elimination. Hence, $d_{k+1}^{(k)}$ is the first entry in the first row of S . But since S is a singular M -matrix with zero column sums (see Lemma 3), this is equal to the negated sum of the entries below the first one in the first column of S , hence (7). \square

It has already been observed that the entries of each of the matrices $Z^{(k)}$ are non-negative. Since $j \geq k + 2$ and since the entries of $z_{k+1}^{(k)}$ past the $(k + 1)$ th one are all zero, the only entries of a_i^T that enter the computation of $d_{k+1}^{(k)}$ via (7) are nonpositive. Hence, all the quantities involved in (7) are nonnegative, and no subtractions are performed.

With this modification, the GTH-like version of the null vector algorithm runs as follows.

Algorithm 2. Null Vector Algorithm (GTH-like variant).

Let $z_i^{(0)} = e_i, i = 1 : n$.

for $k = 1 : n - 1$

$$d_k^{(k-1)} = - \sum_{i=k+1}^n a_i^T z_k^{(k-1)}$$

for $j = k + 1 : n$

$$d_j^{(k-1)} := a_k^T z_j^{(k-1)}$$

$$z_j^{(k)} := z_j^{(k-1)} - \left(\frac{d_j^{(k-1)}}{d_k^{(k-1)}} \right) z_k^{(k-1)}$$

end

end

A few observations are in order. In practice, the algorithm is implemented using only the probabilities $p_{ij}, i \neq j$. The update formula can be written as

$$z_j^{(k)} := z_j^{(k-1)} + \left(\frac{-d_j^{(k-1)}}{d_k^{(k-1)}} \right) z_k^{(k-1)}.$$

In an implementation using only the transition probabilities, the minus sign in the above expression would be redundant. Since $d_k^{(k-1)} > 0$, $-d_j^{(k-1)} \geq 0$, and the entries in the vectors $z_i^{(k-1)}$ are all nonnegative, no negative quantities occur. The algorithm involves no subtractions. For this reason, it can be expected to compute the stationary vector with accuracy similar to that of the GTH method.

Just as GTH is slightly more expensive than Gaussian elimination, so is Algorithm 2 relative to Algorithm 1. The overhead consists of the extra additions needed to calculate the pivots according to (7). An efficient way to do this is by keeping a running sum of the elements of a_k^T , $k = 2, 3, \dots, n$, that contribute to the pivot. Note that with Algorithm 2, it is not possible to generate the rows of A one by one in the course of the algorithm. Also note that the last row of A is no longer redundant. There are several possible implementations of Algorithm 2, some of which trade storage space for arithmetic operations. It is also easy to develop variants that work on A^T rather than A . The choice of implementation, in the end, is likely to be dictated by the storage scheme used or by the order in which the data becomes available.

6. Numerical experiments

The main purpose of this section is to provide some evidence that the GTH-like variant of the direct projection method (Algorithm 2) computes stationary vectors to high accuracy. In the absence of a formal error analysis, numerical experiments on a set of standard test problems are used to support this statement. The algorithms tested include Gaussian elimination (denoted GE in the tables), the original direct projection method (DPM), the GTH algorithm (GTH), and the GTH-like variant of DPM (SDPM). For completeness, results for the QR factorization of A^T (denoted by QR) are also included. All the experiments were computed using double-precision IEEE arithmetic (approximately 16 decimal digits) in MATLAB.

6.1. Test problem 1

The first problem is the well known 8×8 Courtois matrix [10]:

$$P = \begin{bmatrix} 0.85 & 0 & 0.149 & 0.0009 & 0 & 0.00005 & 0 & 0.00005 \\ 0.1 & 0.65 & 0.249 & 0 & 0.0009 & 0.00005 & 0 & 0.00005 \\ 0.1 & 0.8 & 0.0996 & 0.0003 & 0 & 0 & 0.0001 & 0 \\ 0 & 0.0004 & 0 & 0.7 & 0.2995 & 0 & 0.0001 & 0 \\ 0.0005 & 0 & 0.0004 & 0.399 & 0.6 & 0.0001 & 0 & 0 \\ 0 & 0.00005 & 0 & 0 & 0.00005 & 0.6 & 0.2499 & 0.15 \\ 0.00003 & 0 & 0.00003 & 0.00004 & 0 & 0.1 & 0.8 & 0.0999 \\ 0 & 0.00005 & 0 & 0 & 0.00005 & 0.1999 & 0.25 & 0.55 \end{bmatrix}.$$

This matrix is nearly uncoupled with degree of coupling $\gamma = 0.001$. The stationary vector, computed in quadruple precision by the GTH algorithm [11], is given by

Table 1
Results for Courtois matrix

Method	Residual	Error
GE	0.208E–16	0.215E–13
DPM	0.833E–16	0.446E–13
QR	0.694E–16	0.683E–13
GTH	0.278E–16	0.518E–14
SDPM	0.624E–16	0.529E–14

$$\pi^T = \begin{bmatrix} 0.8928265275450187E - 01 \\ 0.9275763750513320E - 01 \\ 0.4048831201636394E - 01 \\ 0.1585331908198259E + 00 \\ 0.1189382069041751E + 00 \\ 0.1203854811060527E + 00 \\ 0.2777952524492734E + 00 \\ 0.1018192664446790E + 00 \end{bmatrix}.$$

The residual $\|\bar{\pi} - \bar{\pi}P\|_1$ and the absolute error $\|\pi - \bar{\pi}\|_1$, where $\bar{\pi}$ denotes the computed stationary vector, are given in Table 1. On this problem GTH and SDPM display comparable accuracy, with an error one order of magnitude smaller than the other methods.

6.2. Test problem 2

The second problem is taken from [11] and is quite similar to one described in [36]. It is actually a parameterized family of 10×10 matrices with varying degree of coupling. Let

$$T = \begin{bmatrix} 0.1 & 0.3 & 0.1 & 0.2 & 0.3 & \beta & 0 & 0 & 0 & 0 \\ 0.2 & 0.1 & 0.1 & 0.2 & 0.4 & 0 & 0 & 0 & 0 & 0 \\ 0.1 & 0.2 & 0.2 & 0.4 & 0.1 & 0 & 0 & 0 & 0 & 0 \\ 0.4 & 0.2 & 0.1 & 0.2 & 0.1 & 0 & 0 & 0 & 0 & 0 \\ 0.6 & 0.3 & 0 & 0 & 0.1 & 0 & 0 & 0 & 0 & 0 \\ \beta & 0 & 0 & 0 & 0 & 0.1 & 0.2 & 0.2 & 0.4 & 0.1 \\ 0 & 0 & 0 & 0 & 0 & 0.2 & 0.2 & 0.1 & 0.3 & 0.2 \\ 0 & 0 & 0 & 0 & 0 & 0.1 & 0.3 & 0.2 & 0.2 & 0.2 \\ 0 & 0 & 0 & 0 & 0 & 0.2 & 0.2 & 0.1 & 0.3 & 0.2 \\ 0 & 0 & 0 & 0 & 0 & 0.1 & 0.7 & 0 & 0 & 0.2 \end{bmatrix}.$$

If $\Delta = \text{diag}(\frac{1}{1+\beta}, 1, 1, 1, 1, \frac{1}{1+\beta}, 1, 1, 1, 1)$, then $P = \Delta T$ is a nearly uncoupled stochastic matrix with degree of coupling $\gamma = \frac{\beta}{1+\beta}$. Two values of β are used.

(a) For $\beta = 10^{-7}$ the stationary vector is given by

$$\pi^T = \begin{bmatrix} 0.1008045195787271E + 00 \\ 0.8012666139606563E - 01 \\ 0.3015519514905696E - 01 \\ 0.6031039029811392E - 01 \\ 0.7926508439180686E - 01 \\ 0.1008045195787271E + 00 \\ 0.1967651659427390E + 00 \\ 0.7003949685919185E - 01 \\ 0.1619417899342436E + 00 \\ 0.1197871768713281E + 00 \end{bmatrix}.$$

(b) For $\beta = 10^{-14}$ the stationary vector is given by

$$\pi^T = \begin{bmatrix} 0.1008045115305868E + 00 \\ 0.8012666301149126E - 01 \\ 0.3015519575701284E - 01 \\ 0.6031039151402568E - 01 \\ 0.7926508598986232E - 01 \\ 0.1008045115305868E + 00 \\ 0.1967651699097019E + 00 \\ 0.7003949827125116E - 01 \\ 0.1619417931991359E + 00 \\ 0.1197871792863454E + 00 \end{bmatrix}.$$

The results are shown in Table 2.

6.3. Test problem 3

The last example is taken from [35], where it was used to illustrate that Gaussian elimination can fail even if the chain is not nearly uncoupled and even if no catastrophic cancellation occurs in forming $A = I_n - P^T$ or in the course of the elimination process. Here P is the tridiagonal stochastic matrix determined (uniquely) by

$$p_{i+1,i} = 0.8, \quad \text{and} \quad p_{i,i+1} = 0.1, \quad \text{for } i = 1, 2, \dots, n - 1.$$

Thus $p_{ii} = 0.1$ for all i except for $p_{11} = 0.9$ and $p_{nn} = 0.2$.

Table 2
Results for test problem 2

Method	$\beta = 10^{-7}$		$\beta = 10^{-14}$	
	Residual	Error	Residual	Error
GE	0.121E-15	0.567E-09	0.146E-15	0.376E-02
DPM	0.163E-15	0.567E-09	0.118E-15	0.572E-02
QR	0.163E-15	0.260E-09	0.135E-15	0.705E-02
GTH	0.694E-16	0.135E-15	0.937E-16	0.246E-15
SDPM	0.833E-16	0.177E-15	0.111E-15	0.205E-15

On this example, already for $n = 20$ the solutions computed by Gaussian elimination and the direct projection method are both affected by large errors, especially in the last components. With both methods, the last entry of the solution vector π is computed as negative. In contrast, both the GTH and the modified direct projection method return solution vectors with components that are exact to machine precision. Note that some of the π 's are tiny: for example, $\pi_{20} = O(10^{-18})$. The exact pivots in both Gaussian elimination and direct projection method are all equal to 0.1, except for the last one which is 0. While both GTH and SDPM compute all the pivots exactly, the computed pivots in GE and DPM lose about one digit of accuracy at each step. As a result, by the 17th step the pivot has no accurate digits. It is worth noting that the last three entries of π returned by the QR algorithm have no accurate digits. Exactly the same results were observed for larger values of n (up to $n = 300$, the largest computed example).

These experiments suggest that Algorithm 2 is as accurate as the GTH algorithm. A formal error analysis will be presented elsewhere.

7. Conclusions

The main conclusion of this paper is that the direct projection method, and especially its GTH-like variant developed in Section 5, is especially well suited for calculations involving Markov chains. These include the determination of the stationary vector, the computation of the group inverse of the generator matrix, and updating these quantities after the transition matrix undergoes a rank-one change (row modification).

For all of these tasks, the direct projection method requires no more operations or storage than the standard techniques based on Gaussian elimination; in some cases it requires less. Moreover, there is recent evidence that the direct projection method is advantageous on parallel machines.

Although a formal round-off error analysis of Algorithm 2 has not been carried out, the numerical evidence suggests that this algorithm computes an approximation

to the stationary vector with low relative error in each component, and is as accurate and robust as the well known GTH algorithm.

In this paper, it has been assumed that the transition matrix P is dense and not too large. When P is very large and sparse, the techniques considered in this paper are generally too expensive in terms of operation count and storage requirements. Nevertheless, they can be adapted to the task of computing effective parallel preconditioners for Krylov subspace methods; see [6].

The direct projection method has a number of interesting features (not just in the Markov chain setting), and it is unfortunate that the algorithm has not received more widespread attention. It is hoped that this paper will contribute to rectify this situation.

Acknowledgements

I would like to thank Carl Meyer for introducing me (more than 10 years ago!) to both themes of this paper, the direct projection method and finite Markov chains. Investigating the application of the former to the latter was a long overdue task. It is a great pleasure being able to finally dedicate this paper to Carl on the occasion of his 60th birthday.

References

- [1] J.L. Barlow, Perturbation results for nearly uncoupled Markov chains with applications to iterative methods, *Numer. Math.* 65 (1993) 51–62.
- [2] M. Benzi, A direct row-projection method for sparse linear systems, Ph.D. Thesis, North Carolina State University, Raleigh, NC, 1993.
- [3] M. Benzi, C.D. Meyer, A direct projection method for sparse linear systems, *SIAM J. Sci. Comput.* 16 (1995) 1159–1176.
- [4] M. Benzi, C.D. Meyer, M. Tũma, A sparse approximate inverse preconditioner for the conjugate gradient method, *SIAM J. Sci. Comput.* 17 (1996) 1135–1149.
- [5] M. Benzi, M. Tũma, A sparse approximate inverse preconditioner for nonsymmetric linear systems, *SIAM J. Sci. Comput.* 19 (1998) 968–994.
- [6] M. Benzi, M. Tũma, A parallel solver for large-scale Markov chains, *Appl. Numer. Math.* 41 (2002) 135–153.
- [7] A. Berman, R.J. Plemmons, *Nonnegative Matrices in the Mathematical Sciences*, Academic Press, New York, 1979, Reprinted by SIAM, Philadelphia, PA, 1994.
- [8] S.L. Campbell, C.D. Meyer, *Generalized Inverses of Linear Transformations*, Pitman, London, 1979, Reprinted by Dover Publishing, New York, 1991.
- [9] K. Chen, C.H. Lai, Parallel algorithms of the Purcell method for direct solution of linear systems, *Parallel Comput.* 28 (2002) 1275–1291.
- [10] P.J. Courtois, *Decomposability: Queuing and Computer System Applications*, Academic Press, New York, 1977.
- [11] T. Dayar, Stability and conditioning issues on the numerical solution of Markov chains, Ph.D. Thesis, North Carolina State University Raleigh, NC, 1994.

- [12] T. Dayar, W.J. Stewart, On the effects of using the Grassmann–Taksar–Heyman method in iterative aggregation–disaggregation, *SIAM J. Sci. Comput.* 17 (1996) 287–303.
- [13] D.K. Faddeev, V.N. Faddeeva, *Computational Methods of Linear Algebra*, W.H. Freeman and Co., San Francisco, CA, 1963.
- [14] R. Fletcher, Dense factors of sparse matrices, in: M.D. Buhmann, A. Iserles (Eds.), *Approximation Theory and Optimization* (Cambridge, 1996), Cambridge University Press, Cambridge, 1997, pp. 145–166.
- [15] G.E. Forsythe, Review of [37], *Mathematical Reviews* # 15,471h (1953).
- [16] L. Fox, H.D. Huskey, J.H. Wilkinson, Notes on the solution of algebraic linear simultaneous equations, *Quart. J. Mech. Appl. Math.* 1 (1948) 253–280.
- [17] R.E. Funderlic, J.B. Mankin, Solution of homogeneous systems of linear equations arising from compartmental models, *SIAM J. Sci. Statist. Comput.* 2 (1981) 375–383.
- [18] R.E. Funderlic, C.D. Meyer, Sensitivity of the stationary vector for an ergodic Markov chain, *Linear Algebra Appl.* 76 (1986) 1–17.
- [19] R.E. Funderlic, R.E. Neumann, R.J. Plemmons, LU decompositions of generalized diagonally dominant matrices, *Numer. Math.* 40 (1982) 57–69.
- [20] R.E. Funderlic, R.J. Plemmons, Updating LU factorizations for computing stationary distributions, *SIAM J. Algebr. Disc. Meth.* 7 (1986) 30–42.
- [21] G.H. Golub, C.D. Meyer, Using the QR factorization and group inversion to compute, differentiate, and estimate the sensitivity of stationary probabilities for Markov chains, *SIAM J. Algebr. Disc. Meth.* 7 (1986) 273–281.
- [22] W.K. Grassmann, M.I. Taksar, D.P. Heyman, Regenerative analysis and steady state distributions for Markov chains, *Oper. Res.* 33 (1985) 1107–1116.
- [23] M.R. Hestenes, E. Stiefel, Methods of conjugate gradients for solving linear systems, *J. Res. Nat. Bur. Standards* 49 (1952) 409–436.
- [24] D.P. Heyman, D.P. O’Leary, What is fundamental for Markov chains: first passage times, fundamental matrices, and group generalized inverses, in: W.J. Stewart (Ed.), *Computations with Markov Chains*, Kluwer Academic Publishers, Boston/London/Dordrecht, 1995, pp. 151–161.
- [25] A.S. Householder, *The Theory of Matrices in Numerical Analysis*, Blaisdell Publishing Co., New York, 1964, Reprinted by Dover Publishing, New York, 1975.
- [26] J.J. Hunter, Stationary distributions of perturbed Markov chains, *Linear Algebra Appl.* 82 (1986) 201–214.
- [27] I.C.F. Ipsen, C.D. Meyer, Uniform stability of Markov chains, *SIAM J. Matrix Anal. Appl.* 15 (1994) 1061–1074.
- [28] C.D. Meyer, Generalized inversion of modified matrices, *SIAM J. Appl. Math.* 24 (1973) 315–323.
- [29] C.D. Meyer, The role of the group generalized inverse in the theory of finite Markov chains, *SIAM Rev.* 17 (1975) 443–464.
- [30] C.D. Meyer, Stochastic complementation, uncoupling Markov chains, and the theory of nearly reducible systems, *SIAM Rev.* 31 (1989) 240–272.
- [31] C.D. Meyer, Sensitivity of the stationary distribution of a Markov chain, *SIAM J. Matrix Anal. Appl.* 15 (1994) 715–728.
- [32] C.D. Meyer, J.M. Shoaf, Updating finite Markov chains by using techniques of group matrix inversion, *J. Statist. Comput. Simul.* 11 (1980) 163–181.
- [33] J. Morris, An escalator process for the solution of linear simultaneous equations, *Philos. Mag.* 7 (1946) 106–120.
- [34] C.A. O’Cinneide, Entrywise perturbation theory and error analysis for Markov chains, *Numer. Math.* 65 (1993) 109–120.
- [35] C.A. O’Cinneide, Relative-error bounds for the LU decomposition via the GTH algorithm, *Numer. Math.* 75 (1996) 507–519.
- [36] C.C. Paige, P.H. Styan, P.G. Wachter, Computation of the stationary distribution of a Markov chain, *J. Statist. Comput. Simul.* 4 (1975) 1156–1179.

- [37] E.W. Purcell, The vector method of solving simultaneous linear equations, *J. Math. Phys.* 32 (1953) 180–183.
- [38] G.W. Stewart, Conjugate direction methods for solving systems of linear equations, *Numer. Math.* 21 (1973) 283–297.
- [39] G.W. Stewart, Gaussian elimination, perturbation theory and Markov chains, in: C.D. Meyer, R.J. Plemmons (Eds.), *Linear Algebra, Markov Chains, and Queueing Models*, Springer, New York, NY, 1993, pp. 59–69.
- [40] G.W. Stewart, G. Zhang, On a direct method for the solution of nearly uncoupled Markov chains, *Numer. Math.* 59 (1991) 1–12.
- [41] W.J. Stewart, *Introduction to the Numerical Solution of Markov Chains*, Princeton University Press, Princeton, NJ, 1994.
- [42] R.L. Tweedie, Perturbations of countable Markov chains and processes, *Ann. Inst. Statist. Math.* 32 (1980) 283–290.
- [43] J.H.M. Wedderburn, *Lectures on Matrices*, Am. Math. Soc. Colloq. Publ. XVII, American Mathematical Society, Providence, RI, 1934, Reprinted by Dover Publishing, New York, 1964.
- [44] G. Zhang, On the sensitivity of the solution of nearly uncoupled Markov chains, *SIAM J. Matrix Anal. Appl.* 14 (1993) 1112–1123.