# ON THE EFFECTS OF USING THE GRASSMANN–TAKSAR–HEYMAN METHOD IN ITERATIVE AGGREGATION–DISAGGREGATION

TUĞRUL DAYAR[1] AND WILLIAM J. STEWART [2]

**Abstract.** Iterative aggregation–disaggregation (IAD) is an effective method for solving finite nearly completely decomposable (NCD) Markov chains. Small perturbations in the transition probabilities of these chains may lead to considerable changes in the stationary probabilities; NCD Markov chains are known to be ill–conditioned. During an IAD step, this undesirable condition is inherited by the coupling matrix and one confronts the problem of finding the stationary probabilities of a stochastic matrix whose diagonal elements are close to 1. In this paper, the effects of using the Grassmann–Taksar–Heyman (GTH) method to solve the coupling matrix formed in the aggregation step are investigated. Then, the idea is extended in such a way that the same direct method can be incorporated into the disaggregation step. Finally, the effects of using the GTH method in the IAD algorithm on various examples are demonstrated, and the conditions under which it should be employed are explained.

**Key words.** Markov chains, decomposability, stationary probability, aggregation–disaggregation, Gaussian elimination, sparsity schemes

**AMS subject classifications.** 60J10, 60J27, 65F05, 65F10, 60–04

## 1 Introduction

NCD Markov chains are irreducible stochastic matrices that can be ordered so that the matrix of transition probabilities has a block structure in which the nonzero elements of the off–diagonal blocks are small compared with those of the diagonal blocks. Such matrices often arise in queueing network analysis, large scale economic modeling, and computer systems performance evaluation. We have

$$(1.1) \qquad P_{n\times n} = \begin{array}{c@{}c} \begin{array}{cccc} n_1 & n_2 & \cdots & n_N \end{array} & \\ \left(\begin{array}{cccc} P_{11} & P_{12} & \cdots & P_{1N} \\ P_{21} & P_{22} & \cdots & P_{2N} \\ \vdots & \vdots & \ddots & \vdots \\ P_{N1} & P_{N2} & \cdots & P_{NN} \end{array}\right) & \begin{array}{c} n_1 \\ n_2 \\ \vdots \\ n_N \end{array} \end{array}$$

The subblocks $P_{ii}$ are square, of order $n_i$, with $n = \sum_{i=1}^{N} n_i$. Let the stationary distribution of $P$, $\pi$ (i.e., $\pi P = \pi$, $\|\pi\|_1 = 1$), be partitioned conformally with $P$ such that $\pi = (\pi_1, \pi_2, \ldots, \pi_N)$. Each $\pi_i$, $i = 1, 2, \ldots, N$, is a row vector having $n_i$ elements. Let $P = \mathrm{diag}(P_{11}, P_{22}, \ldots, P_{NN}) + E$. The quantity $\|E\|_\infty$ is referred to as the *degree of coupling* and it is taken to be a measure of the decomposability of the matrix (see [5]). If it were zero, then $P$ would be reducible.

NCD Markov chains that appear in applications are quite large and sparse, possibly having more than thousands of states. For such large chains, direct methods can generate immense fill–in during

[1] The work of this author was performed while the author was at North Carolina State University, Raleight, NC. Current address: Computer Engineering and Information Science Department, Bilkent University, 06533 Bilkent, Ankara, Turkey (tugrul@bilkent.edu.tr).

[2] Department of Computer Science, North Carolina State University, Raleigh, NC 27695–8206, USA (billy@markov.csc.ncsu.edu).

the triangularization phase of the solution process, and due to storage limitations they become impractical. Moreover, when the system is nearly completely decomposable, there are eigenvalues close to 1, and the poor separation of the unit eigenvalue implies a slow rate of convergence for standard matrix iterative methods [12]. IAD methods do not suffer from these limitations [7], [23].

The idea in IAD methods is to observe the system in isolation in each of the diagonal blocks as if the system is completely decomposable (see [15]) and to compute the stationary probability distribution of each diagonal block. However, there are two problems with this approach. First, since the diagonal blocks are substochastic, the off–diagonal probability mass must somehow be incorporated into the diagonal blocks. Second, the probabilities obtained by this approach are conditional probabilities, and this condition has to be removed by weighing each probability subvector by the probability of being in that group of states. Only if these two problems are overcome can one form the stationary distribution of the Markov chain by weighing the subvectors and pasting them together.

For the transition probability matrix $P$, the stochastic complement of $P_{ii}$ [8], is

$$S_{ii} = P_{ii} + P_{i*}(I - P_i)^{-1} P_{*i},$$

where

$P_{i*}$ : $n_i \times (n - n_i)$ matrix composed of the $i^{th}$ row of blocks of $P$ with $P_{ii}$ removed,

$P_{*i}$ : $(n - n_i) \times n_i$ matrix composed of the $i^{th}$ column of blocks of $P$ with $P_{ii}$ removed,

$P_i$ : $(n - n_i) \times (n - n_i)$ principal submatrix of $P$ with $i^{th}$ row and $i^{th}$ column of blocks removed.

Each stochastic complement is the stochastic transition probability matrix of a smaller irreducible Markov chain obtained by observing the original process in the corresponding block of states. The conditional stationary probability vector of the $i^{th}$ block is $\pi_i/\|\pi_i\|_1$ and it may be computed by solving $(\pi_i/\|\pi_i\|_1)S_{ii} = \pi_i/\|\pi_i\|_1$ (see [8] for details). However, each stochastic complement has an embedded matrix inversion which may require excessive computation. An alternative solution technique is to approximate $S_{ii}$ by accumulating the off–diagonal mass $P_{i*}$ into the diagonal block $P_{ii}$ on a row by row basis. There are various ways in which this can be done [21]. Thereafter, an approximation of the conditional stationary vector of the block of states may be found by solving the corresponding linear system as before.

To determine the probability of being in a certain block of states, one needs to construct the so–called coupling matrix which shrinks each block down to a single element, forming an $N \times N$ irreducible stochastic matrix. This is accomplished by first replacing each row of each block by the sum of elements in that block row. The sum of elements of row $k$ of $P_{ij}$ gives the probability of leaving state $k$ of block $i$ and entering into block $j$. Therefore, the operation to be performed for each block is $P_{ij}e$. In what follows, $e$ is a column vector of 1's whose length is determined by the context in which it is used. Moreover, each column vector $P_{ij}e$ must be reduced to a scalar. The total probability of leaving block $i$ to enter into block $j$ may be determined by summing the elements of $P_{ij}e$ after each element of this vector has been multiplied by the probability of being in that state (given that the system is in block $i$). These multiplicative constants may be obtained from the stationary vector elements; they are the components of $\pi_i/\|\pi_i\|_1$. Hence, the $ij^{th}$ element of the coupling matrix is given by $c_{ij} = (\pi_i/\|\pi_i\|_1)P_{ij}e$. The stationary vector of the coupling matrix gives the stationary probability of being in each one of the block of states. In other words, the weighing factors mentioned before are the elements of the stationary vector of the coupling matrix. However, the stationary vector itself is needed to form the coupling matrix. Since the aim is to compute the stationary vector, one can approximate the coupling matrix by using an approximate stationary vector and improve on the approximate solution iteratively [21].

Without further ado, the IAD algorithm is provided. Convergence analysis of the algorithm appears in [17]. Additional discussion may be found in [21] and [2].

**The Generic Iterative Aggregation-Disaggregation (IAD) Algorithm:**

1. Let $\pi^{(0)} = (\pi_1^{(0)}, \pi_2^{(0)}, \ldots, \pi_N^{(0)})$ be a given initial approximation of the solution. Set $k = 1$.
2. Construct the coupling matrix $C^{(k-1)}$
$$c_{ij}^{(k-1)} = \frac{\pi_i^{(k-1)}}{\|\pi_i^{(k-1)}\|_1} P_{ij} e.$$
3. Solve the eigenvector problem
$$\xi^{(k-1)} C^{(k-1)} = \xi^{(k-1)}, \quad \|\xi^{(k-1)}\|_1 = 1$$
for $\xi^{(k-1)} = (\xi_1^{(k-1)}, \xi_2^{(k-1)}, \ldots, \xi_N^{(k-1)})$.
4. (a) Compute the row vector
$$z^{(k)} = (\xi_1^{(k-1)} \frac{\pi_1^{(k-1)}}{\|\pi_1^{(k-1)}\|_1}, \xi_2^{(k-1)} \frac{\pi_2^{(k-1)}}{\|\pi_2^{(k-1)}\|_1}, \ldots, \xi_N^{(k-1)} \frac{\pi_N^{(k-1)}}{\|\pi_N^{(k-1)}\|_1}).$$
(b) Solve the $N$ systems of equations
$$\pi_i^{(k)} = \pi_i^{(k)} P_{ii} + \sum_{j>i} z_j^{(k)} P_{ji} + \sum_{j<i} \pi_j^{(k)} P_{ji}$$
for $\pi_i^{(k)}, \quad i = 1, 2, \ldots, N$.
5. Test $\pi_i$ for convergence. If the desired accuracy is attained, then stop and take $\pi_i$ as the stationary probability vector of $P$. Else set $k = k + 1$ and go to step 2.

In the IAD algorithm, steps 2 and 3 form the aggregation step and 4(b), which is essentially a block Gauss–Seidel iteration, forms the disaggregation step. In step 2, $\pi_i^{(k-1)}/\|\pi_i^{(k-1)}\|_1$, is an approximation of the stationary distribution of the stochastic complement of $P_{ii}$. The weighing factors, $(\|\pi_1\|_1, \|\pi_2\|_1, \ldots, \|\pi_N\|_1)$, are approximated by $\xi^{(k-1)}$ in step 3. Each iteration of the IAD algorithm reduces the residual error (i.e., $\|\pi(I - P)\|$) by a factor of $\|E\|$ (see [14], p. 80).

The next section discusses how a modified version of Gaussian elimination may be used to enforce stability in the solution of the coupling matrix. In §3, the idea is extended so that it can be used in a nonsingular system of equations with a substochastic coefficient matrix. Section 4 discusses certain implementation issues, and §5 provides numerical experiments with the IAD algorithm on NCD Markov chains.

## 2 Solving the Coupling Matrix

The coupling matrix is an irreducible stochastic matrix of order N; that is, each of its two dimensions is equal to the number of blocks in the NCD Markov chain, and all states form a single communicating class. The goal is to solve the singular system $\xi^{(k)}C^{(k)} = \xi^{(k)}$ subject to $\|\xi^{(k)}\|_1 = 1$, which is termed the normalization equation. Essentially, each row of $C^{(k)}$ is a linear combination of the others, and inclusion of the normalization equation enables one to replace the redundant equation, thus achieving full rank in the set of equations. Being an irreducible stochastic matrix, the coupling matrix has a unique unit eigenvalue. All other $N-1$ eigenvalues are close to 1. The distance of these other eigenvalues to 1 depends on the degree of coupling.

A careful inspection reveals that one can solve the equivalent tansposed system

$$(2.1) \qquad (I - C^{(k)})^T(\xi^{(k)})^T = 0, \qquad \|\xi^{(k)}\|_1 = 1.$$

The justification for considering this form of the problem is the following. (2.1) is the conventional form in which a (homogeneous) linear system of equations is expressed: the coefficient matrix postmultiplied by the unknown vector equals the (zero) right–hand side. Besides, there are already existing algorithms that may be used with this form of a linear system. But most importantly, as it is explained later in §4, a row–wise sparse storage implementation will spend extra time in the substitution phase of the non–transposed system of equations.

$(I - C^{(k)})^T$ is a singular M–matrix (see [1]) with 0 column sums, and the unique null vector of unit 1–norm is sought. For such a matrix, Gaussian elimination (GE) preserves column diagonal dominance throughout its computation so that the multiplier element at each step is bounded by 1 thereby eliminating the need for pivoting. This follows from the fact that the pivot element at a given step has the largest magnitude among all elements that lie in the unreduced part of its respective column. Note that the same argument would be valid for the original system of equations if $C^{(k)}$ were doubly–stochastic.

It is well known that since $C^{(k)}$ is a perturbation of the identity matrix, all its nonunit eigenvalues are close to 1. Due to this fact, iterative methods tend to converge slowly. On the other hand, certain stability issues need to be addressed if direct methods are used. As it is shown next, ordinary GE is not stable in the presence of rounding errors on a coupling matrix whose diagonal elements are close to 1.

*Example.* Consider the following irreducible stochastic matrix:

$$P = \begin{pmatrix} \frac{1}{3} & \frac{2}{3} - \alpha & \alpha \\ \frac{3}{4} - \alpha & \frac{1}{4} & \alpha \\ \alpha & \alpha & 1 - 2\alpha \end{pmatrix}.$$

Assuming $\alpha$ is small, $P$ is clearly NCD. Additionally, the degree of coupling is $2\alpha$. Now, suppose $2\alpha$ is less than machine epsilon (smallest representable positive floating–point number $\epsilon$ such that $1 + \epsilon > 1$). Let $\pi^{(k)} > 0$. Then, the coupling matrix is rounded to

$$C^{(k)} = \begin{pmatrix} 1 & \alpha \\ 2\alpha & 1 \end{pmatrix}.$$

Note that the coupling matrix is not stochastic. Therefore, the matrix $(I - C^{(k)})^T$ is rounded to

$$(I - C^{(k)})^T = \begin{pmatrix} 0 & -2\alpha \\ -\alpha & 0 \end{pmatrix}.$$

Hence, it is necessary to apply some sort of pivoting strategy in order to proceed with GE. Otherwise, the algorithm breaks down contrary to the general belief that GE applied to stochastic matrices is always stable. Before going any further, this issue should be explained some more.

It is possible for an irreducible NCD Markov chain ordered as described in (1.1) to have zero blocks. Consequently, the coupling matrix of such a chain at any iteration has zeros in locations corresponding to zero blocks in the transition probability matrix. However, the zeros in the coupling matrix occur in just the right places so that the coupling matrix is still irreducible ([8]). Keeping this in mind, the situation that causes GE to break down occurs when the pivot element at a given step of GE is zero in the matrix $(I - C^{(k)})^T$ during the $k^{th}$ iteration of the IAD algorithm. A sufficient condition for the failure of GE during the aggregation step is to have a transition probability matrix with a degree of coupling less than machine epsilon. In this case, all diagonal elements in $(I - C^{(k)})^T$ will be zero, as in the example, and GE will fail in the first step.

The advocated approach to finding a remedy for this situation is the GTH algorithm described in [4] and the direct method discussed in [16]. The original GTH algorithm emerges from probabilistic arguments, and it is shown that the stationary distribution of a Markov chain can be calculated using only nonnegative numbers and avoiding subtraction operations. This algorithm achieves significantly greater accuracy than other algorithms described in the literature [11], [5] since there is no loss of significant digits due to cancellation [6]. Interestingly, the inspiration for the algorithm presented in [16], which is specifically for the solution of NCD Markov chains, is the GTH algorithm. In a recent paper [10], it is shown that the stationary probability vector of an $N \times N$ irreducible Markov chain stored in floating–point form is close to its exact stationary vector. In other words, even if one has an algorithm that does not introduce any errors by itself, the best that can be done is to compute the stationary vector of an $N \times N$ Markov chain with an entrywise relative error of only about $2Nu$ of the exact stationary vector. In fact, the relative error incurred by each element of the floating–point stationary vector is about $2Nu$, where $u$ is the unit roundoff (i.e., the maximum relative error in approximating a real number by its nearest floating–point number). This result follows from the fact that the perturbation introduced by storing the matrix in floating–point form is of order unit roundoff. Moreover, in the same paper it is proven that, if GTH is employed in the solution process of this $N \times N$ Markov chain, the relative error in each entry of the stationary vector will be of order $N^3u$. It should be remarked that the entrywise relative error in the stationary vector for the GTH algorithm is independent of the structure of the matrix and the magnitude of its elements.

The following lemma shows that the GTH way of calculating the pivot element may also be applied to the transposed system of equations (2.1). It is this form of the equations in which multipliers are bounded by 1, and it will be demonstrated by numerical experiments that the type of implementation chosen (i.e., transposed versus nontransposed system of equations) has no effect on the accuracy of the results obtained.

LEMMA. Let $A = I - P^T$, where $P$ is a stochastic matrix, and

$$A = \begin{pmatrix} a_{11} & v^T \\ u & A_{22} \end{pmatrix}.$$

Then $e^T A = 0$ and $e^T u = -a_{11}$. If

$$L_1 A = \begin{pmatrix} a_{11} & v^T \\ 0 & \tilde{A}_{22} \end{pmatrix},$$

where

$$L_1 = \begin{pmatrix} 1 & 0 \\ -u/a_{11} & I \end{pmatrix} \qquad L_1^{-1} = \begin{pmatrix} 1 & 0 \\ u/a_{11} & I \end{pmatrix},$$

then $\tilde{A}_{22}$ (like $A$) is a singular M–matrix having 0 column sums.

*Proof.* We have

$$e^T A = e^T (L_1^{-1} L_1) A = 0$$

$$e^T L_1^{-1} = (1, 1, \ldots, 1) \begin{pmatrix} 1 & 0 \\ u/a_{11} & I \end{pmatrix} = (0, 1, \ldots, 1)$$

$$(0, 1, \ldots, 1)(L_1 A) = 0 \quad \Rightarrow \quad e^T \tilde{A}_{22} = 0$$

Other steps may be shown similarly. Note that the lemma is also valid for the negation of any generator matrix. □

The properties of a singular M–matrix coupled with the GTH idea of avoiding subtractions and negative numbers suggests the following modification to GE. At each step of GE, rather than calculating the pivot element in the usual way, one can correct the pivot by replacing it with the negated sum of the off–diagonal elements in the unreduced part of the same column as the pivot. When one mentions the GTH method, it is this approach used in calculating the pivot elements that is implied.

## 3 Using the GTH Method in the Disaggregation Step

In a given iteration, the disaggregation step of the IAD method uncouples the NCD Markov chain to obtain a new estimate for the stationary distribution. As indicated in [21], in order to achieve an even better approximation of $\pi$, at the $(k+1)^{st}$ iteration of the IAD algorithm, one solves

$$(I - P_{ii}^T)(\pi_i^{(k+1)})^T = b_i, \tag{3.1}$$

where $\pi_i^{(k+1)}$ is the $(k+1)^{st}$ approximation of $\pi_i$ and $b_i^T = \sum_{j<i} \pi_j^{(k+1)} P_{ji} + \sum_{j>i} z_j^{(k+1)} P_{ji}$ for $i = 1, 2, \ldots, N$ (see step 4(b) of the IAD algorithm).

$P_{ii}$ is a strictly substochastic matrix of order $n_i$, and $b_i \neq 0$, which gives a nonhomogeneous system of linear equations with nonsingular coefficient matrix. Had the coefficient matrix been stochastic, the same GTH technique that is utilized in solving the coupling matrix could be clearly employed. In order to be able to use GTH in solving the diagonal block system, first certain modifications must be made. These modifications involve adding one more equation and augmenting the matrix with $b_i$ to put the system into the form

$$W_i^T (\pi_i^{(k+1)}, \hat{\pi}_i^{(k+1)})^T = 0, \tag{3.2}$$

where

$$W_i^T = \begin{array}{c} \\ \\ \end{array}\begin{array}{cc} n_i & 1 \\ \left(\begin{array}{c} I - P_{ii}^T \\ w_i^T \end{array}\right. & \left.\begin{array}{c} b_i \\ \hat{w}_i \end{array}\right) \end{array} \begin{array}{c} \\ n_i \\ 1 \end{array},$$

$w_i = P_{i*}e$, $\hat{w}_i = -b_i^T e$, and $\hat{\pi}_i^{(k+1)}$ is introduced so that the solution vector has as many columns as the coefficient matrix. In other words, $(w_i^T, \hat{w}_i)$ sums up the columns of $W_i^T$ to 0. The values of $\hat{w}_i$ and $\hat{\pi_i}^{(k+1)}$ are irrelevant, because just as in (2.1), one has a singular system with $n_i + 1$ equations, and after $W_i^T$ is reduced to upper–triangular form, the last row will be all 0's. Hence, the reduction needs to be carried out for only $n_i$ steps. What needs to be done for the computation of $(\pi_i^{(k+1)})^T$ is to use the first $n_i$ elements of column $n_i + 1$ in the upper–triangular matrix as the right–hand side in the back substitution phase.

It is not possible to put GE to use in the disaggregation step in the previous example in §2 or in similar problems on the diagonal blocks for which $\|P_{i*}\|_\infty < \epsilon$. Under the given condition, $I - P_{ii}^T$ is a singular matrix in floating–point form; however, the vector $b_i$ is still nonzero, and therefore the system of equations in (3.2) is inconsistent. On the other hand, GTH computes $w_i^T$ in $W_i^T$ by using the nonzero elements in $P_{i*}$ and forms the pivot by summing offdiagonal entries in the unreduced part of the same column as the pivot. Hence, GTH may be applied to solve such blocks.

## 4 Implementation Considerations

As mentioned before, NCD Markov chains that arise in real–life applications are generally large and sparse. This necessitates the design and employment of sparse storage schemes, which essentially store only the nonzero elements of the transition probability/rate matrix.

So far, the application of the GTH (and GE, for that matter) algorithm to the systems (2.1) and (3.1) is considered. (3.1) is written in the form of (3.2) so that it has a singular M–matrix with 0 column sums as its coefficient matrix just as in (2.1). Now, the alternative nontransposed systems of equations may be written. The system that corresponds to (2.1) is given by

$$(4.1) \qquad \xi^{(k)}(I - C^{(k)}) = 0, \qquad \|\xi^{(k)}\|_1 = 1.$$

Similarly, the equivalent of (3.2) is

$$(4.2) \qquad (\pi_i^{(k+1)}, \hat{\pi_i}^{(k+1)})W_i = 0.$$

Define

*Scheme 1:* The transposed systems of equations in (2.1) and (3.2).

*Scheme 2:* The nontransposed systems of equations in (4.1) and (4.2).

Unless otherwise specified, reductions mean row–reductions (i.e., the addition of a multiple of a lower–indexed row to a higher–indexed row) in a given system of equations. The last assumption is that the coefficient matrices are supplied to both schemes in the nontransposed version. This is a fairly reasonable assumption, since the matrices are usually generated by following the possible transitions from a given state, implying a row–wise generation.

The advantage of reducing the coefficient matrices in Scheme 1 rather than the ones in Scheme 2 to upper–triangular form is twofold:

- the multiplier elements at each step of the reduction are bounded by 1 and
- only the upper–triangular matrix needs to be stored during the reduction process.

Although the multipliers in Scheme 2 are not necessarily bounded by 1, the growth factor still cannot be greater than 1 as indicated in [22]. However, if row–reductions are carried out in Scheme 2, both the upper–triangular matrix and the lower–triangular matrix, which contains the multipliers, have to be stored during the triangularization process. The reason is that if row–reductions are performed on the nontransposed coefficient matrix, the $LU$ decomposition will provide a nonsingular lower–triangular matrix and a singular upper–triangular matrix that has 0's in the last row. Although the decomposition is premultiplied by the stationary vector, one only has to carry out a single substitution phase (see [9], p.724) just as in the transposed system of equations (but this time involving the lower–triangular matrix), thereby making it necessary to store both the lower and the upper–triangular matrix during the reduction. One cure that comes to mind is to store only the upper–triangular matrix during the reduction process and to reconstruct the multiplier matrix after the reduction is over. Though quite straightforward, this solution is inefficient due to the superfluous operations performed in calculating the multipliers for the second time. On the other hand, Scheme 1 calls for the transposition of the coefficient matrix before executing GE or GTH.

What follows is a discussion of the effects of the two implementation schemes on a *semisystematic row–wise sparse storage format* (see [13]). A row–wise sparse storage format is a data structure that stores the nonzero elements in the coefficient matrix row by row. Semisystematic means the elements of row $i$ precede those of row $i+1$, but the elements within a given row need not be ordered. Together with the real value of each nonzero element, the column index of the element is stored as well. Hence, two arrays of the same length, one real and the other integer, are needed. Last, in order to have access to the rows, the starting location of each row has to be stored in an integer array. To facilitate the computation of the number of nonzero elements per row, an extra element whose value is equal to the total number of nonzero elements plus 1 is appended to this array.

The GE algorithm, no matter which scheme is chosen, may be implemented using delayed updates, which means at step $k$ all reductions on row $k+1$ may be carried out and the row compacted and stored before the algorithm continues in the next step with the update of the next row. The property of the GE algorithm which makes this efficient implementation possible is that reduction on a given row requires the addition of appropriate multiples of lower–indexed rows to it. Hence, there is no data dependency between a row to be reduced and higher–indexed rows in GE. The importance of this idea lies in the fact that, it is sufficient to fully expand only the row to be updated at a given step, whereas all other rows with higher indices can still be held in compact form. However, this is no longer applicable when GTH is used in Scheme 1, since all updates due to a row must be finished before the algorithm proceeds with the next step. This situation is dictated by the new way of calculating the pivot elements. In the GTH algorithm, the pivot element is formed by summing the most recently updated elements that lie below the pivot, which implies a dependency of data between the pivot row and the higher–indexed rows. Since storage is limited, one possibility is to expand each row that is operated on, update it, and finally store it back in the appropriate location. The drawback is that this sequence of operations on a row, depending on the fill–in, most probably causes other elements in sparse storage to be shifted around the data structure (extra nonzero entries are likely to occur after a row update). Consequently, it is safe to assume that GTH, with such an implementation, spends more time doing memory reads and writes for larger chains. As pointed out in §2, there is a slight difference in the way the single substitution phase in Schemes 1 and 2 is implemented in the GTH algorithm when a row–wise sparse storage format is used. In the substitution phase, Scheme 2 calls for the solution of a homogeneous linear system in which the stationary vector is postmultiplied by the lower–triangular matrix obtained ¿from the $LU$ decomposition. The substitution is accomplished by accessing a different column of the lower–triangular matrix at each step. This situation makes it necessary to have a doubly–nested loop in the code for accessing the elements of the columns. On the other hand, the substitution phase of Scheme 1 requires the solution of a homogeneous linear system in which the stationary vector is premultiplied by an upper–triangular matrix. This being so, the substitution phase of Scheme 1 conforms nicely to a row–wise sparse storage implementation, and, therefore, it may be handled in a single loop that accesses a row of the upper–triangular matrix at each step. In the actual implementation, the normalization equation (i.e., $\|\boldsymbol{\pi}\|_1 = 1$) is used to discard the last equation and form a nonzero right–hand side in both schemes.

A suggestion made by G. W. Stewart ([18]) as a compromise between the high accuracy of GTH and the implementation efficiency of GE is the following. Rather than performing a correction on the pivot element at each step of GTH, the suggestion is that a pivot should be corrected only when cancellation would produce an inaccurate pivot. For the systems arising in the IAD algorithm, cancellation seems to occur at quite predictable places. Specifically, for the coupling matrix $C^{(k)}$, cancellation occurs in the computation of the diagonal elements of $I - C^{(k)}$. The suggested cure is to compute the diagonal elements of $I - C^{(k)}$ from the off–diagonal elements at the start and to use GE thereafter. For the solution of the diagonal blocks, cancellation is most likely to occur in the computation of the last pivot element. In this case the recommendation is that GE should be applied to the augmented system in (3.2) (or (4.2)), and only the last pivot should be computed by summing the off–diagonal probability mass as in GTH. It is expected that using the suggested scheme in the IAD algorithm will result in a competitive solver (with the GTH method in terms of relative accuracy and with the GE method in terms of total time spent in the IAD algorithm). Results of experiments with this scheme are discussed in §5.

An alternative storage scheme is one that keeps the unused array elements in the form of a linked list (basically, a freepool). In this case, an extra integer array, which, for each nonzero element, stores the location of the next nonzero element in the same row, is required. If a given element is the last in its row, then the corresponding pointer may be set to 0. Though this scheme circumvents the problem of moving array elements back and forth in memory, it introduces overhead due to the manipulation of the freepool. Note that just as in the ordinary row–wise sparse storage format, all elements in a row may be kept ordered according to their column indices in this last implementation. Finally, it should be pointed out that the effect of using a column–wise sparse storage format with Scheme 2 is the same as using a row–wise sparse storage format with Scheme 1 and vice versa.

Although, not as costly as the extra reads and writes due to the discussed implementation problems, GTH has $N - k$ extra additions at step $k$ of the elimination procedure, amounting to $\sum_{k=1}^{N-1}(N-k) = N(N-1)/2$ extra operations in the solution of the coupling matrix. Similarly, there are $n_i(n_i - 1)/2$ extra additions in the solution of the $i$th diagonal block. Clearly, this is an upper bound. Since these extra addition operations are performed only on nonzero elements in a sparse storage implementation, the actual overhead may be smaller. Moreover, because of the need to introduce one more row to the linear system, extra $n_i + 1$ locations are used during the solution of each $P_{ii}$ in the disaggregation step. Last, in order to form the $(n_i + 1)^{st}$ equation during the disaggregation step, a number of additions equal to the number of nonzero elements in $P_{i*}$ are performed. If extra memory locations are available, these $N$ matrix–vector multiplications may be performed at the outset and the resulting vectors stored for future use.

Note that both programming difficulties and overhead storage (arrays to store the indices of nonzero array elements and pointers between elements) increase with the sophistication of the storage scheme. Together with the row–wise sparse storage format, experiments with the sparse storage format proposed by Knuth (see [13] for a brief explanation), which provides equally fast access to columns and rows of a matrix, are conducted. For each nonzero element, in addition to having a pointer to the next nonzero element in the same row, there is an additional pointer to the next nonzero element in the same column. Obviously, there are two arrays that keep the locations of the first nonzero element in each row and each column, respectively. However, as remarked earlier, extra time will be spent for obtaining memory elements from and returning memory elements to the freepool with this kind of implementation. With Knuth's sparse storage format, there is no need to transpose the matrix because rows may be treated as columns and columns as rows. Although one may be inclined to think that Knuth's sparse storage format is more advantageous than the row–wise sparse storage format, due to the overhead of handling the freepool, numerical experiments suggest that Knuth's format is likely be useful only if implemented in a programming language that provides primitives for pointers to memory.

If some information about the structure of the coefficient matrix is available, then another possibility may be to employ a two–dimensional banded storage structure. As for GE, the row–wise sparse storage format with Scheme 1 is recommended.

Keeping in mind that memory is much slower than CPU, one must evaluate the significance of extra operations performed and excess time spent in a sparse storage implementation as opposed to the increased stability and accuracy that accrue from GTH in the context of real–life applications.

## 5 Numerical Results

Experiments with the IAD algorithm are carried out in sparse storage on a SUN SPARC station 2. All routines used are part of the software package MARCA (Markov Chain Analyzer) (see [20]). The routines are written in FORTRAN and compiled in both double–precision and quadruple–precision floating–point arithmetic. For each problem solved, the residual error and the relative error in the solution are computed. The relative error is computed as $\|\pi - \tilde{\pi}\|_2 / \|\pi\|_2$, where $\pi$ is the quadruple–precision solution and $\tilde{\pi}$ is the double–precision solution obtained by the IAD algorithm. Both $\pi$ and $\tilde{\pi}$ are normalized so that their 1–norms are unity. The residual error is computed as $\|(I - P^T)\tilde{\pi}^T\|_2$, which theoretically is 0 if $\pi = \tilde{\pi}$. See Table 1.

The problem associated with GE, when NCD Markov chains with a degree of coupling less than machine epsilon are to be solved, has already been shown. Therefore, the emphasis is on problems with a degree of coupling larger than machine epsilon in order to see how GE and GTH behave

TABLE 1
*Notation for Parameters of Numerical Methods.*

| | |
|---|---|
| $n$ | Order of the stochastic transition matrix, $P$ |
| $n_z$ | Number of nonzero elements in the matrix |
| $N$ | Number of strongly connected components |
| $nzb$ | Number of nonzero blocks |
| $Iter$ | Number of iterations required to get a residual norm of less than $10^{-15}$ |
| $T_{total}$ | Total time spent in the IAD algorithm, (in CPU seconds) |
| $Err_{res}$ | $\|(I - P^T)\tilde{\pi}^T\|_2$ |
| $Err_{rel}$ | $\|\pi - \tilde{\pi}\|_2 / \|\pi\|_2$ |
| $\gamma$ | Decomposability parameter |
| $M_{agg}$ | Method used in aggregation phase |
| $M_{disagg}$ | Method used in disaggregation phase |

TABLE 2
*Solvers Used.*

| | |
|---|---|
| $GE_1$ | Sparse Gaussian elimination using Scheme 1 |
| $GE_2$ | Sparse Gaussian elimination using Scheme 2 |
| $GTH_2$ | Sparse Grassmann–Taksar–Heyman algorithm using Scheme 2 |
| $GTH_1$ | Sparse Grassmann–Taksar–Heyman algorithm using Scheme 1 |

comparatively in this case. Experiments are performed with implementation Schemes 1 and 2 on the row–wise sparse storage format used in MARCA. The GTH implementation of Scheme 1 is accomplished by shifting contents of arrays rather than using the alternative with the freepool. See Table 2.

The first problem appears quite frequently in the literature. Although the order of the matrix considered in this problem is quite small, it is still instructive to examine the effects of using the IAD algorithm on such a problem. The second problem investigated is a real–life example and had been studied with a variety of parameters in the past. The scheme suggested by G. W. Stewart [18] in §4 is implemented, and the outcome of the related experiments are discussed following the presentation of the numerical results.

The decomposability parameter $\gamma$ (in Table 1) is input by the user; it is used to determine the strongly connected components in the transition matrix by simply ignoring the elements that are less than the suggested value. Therefore, $\gamma$ may be taken as an approximation of the degree of coupling. If the matrix is not already in the form (1.1), then symmetric permutations are used to put it into the form in which the diagonal blocks form the strongly connected aggregates.

## 5.1 Test Problem 1

The first problem that is considered is the 8 × 8 Courtois matrix [3] with all row sums equal to 1. The degree of coupling for this matrix is 0.001.

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

$$\pi^T = \begin{pmatrix} 0.8928265275450187E-01 \\ 0.9275763750513320E-01 \\ 0.4048831201636394E-01 \\ 0.1585331908198259E+00 \\ 0.1189382069041751E+00 \\ 0.1203854811060527E+00 \\ 0.2777952524492734E+00 \\ 0.1018192664446740E+00 \end{pmatrix}.$$

Table 3

*Results for Problem 1:* $n = 8, n_z = 41, \gamma = 0.001, N = 3, nzb = 9.$

| $M_{agg}$ | $M_{disagg}$ | *Iter* | $T_{total}$ | $Err_{res}$ | $Err_{rel}$ |
|---|---|---|---|---|---|
| $GE_1$ | $GE_1$ | 4 | 0.04 | $0.286E-16$ | $0.824E-13$ |
| $GE_2$ | $GE_2$ | 4 | 0.04 | $0.347E-16$ | $0.904E-13$ |
| $GTH_2$ | $GTH_2$ | 4 | 0.04 | $0.250E-16$ | $0.420E-15$ |
| $GTH_1$ | $GTH_1$ | 4 | 0.04 | $0.208E-16$ | $0.282E-15$ |

## 5.2 Test Problem 2

The second problem considered appears in [19]. It represents a time–shared, multiprogrammed, paged, virtual memory computer system modeled as a closed queueing network (see Fig. 1).

The system consists of a number of users using the terminals, a central processing unit (CPU), a secondary memory device (SMD), and a filing device (FD). The degree of multiprogramming in the system at any given time is $\eta = \eta_0 + \eta_1 + \eta_2$, where $\eta_0$, $\eta_1$, $\eta_2$ are, respectively, the number of processes in the CPU, SMD, and FD queues at that moment. Furthermore, $\eta_t$ represents the number of processes currently being generated at the terminals but not yet transmitted to the CPU for execution. $\lambda$ is the mean generation time of processes by users working on the terminals. The mean service rate of the CPU depends on $\eta$ and is given by $\mu_0(\eta)$. The processes that leave the CPU proceed to the SMD, to the FD, and back to the terminals with probabilities $p_1$, $p_2$, $1 - p_1 - p_2$, respectively. The mean service rate at the SMD is given by $\mu_1$ and at the FD by $\mu_2$. All arrival and service rates are exponential. Note that, it is not possible to apply analytical techniques to solve the above model since the CPU execution time depends on the degree of multiprogramming.

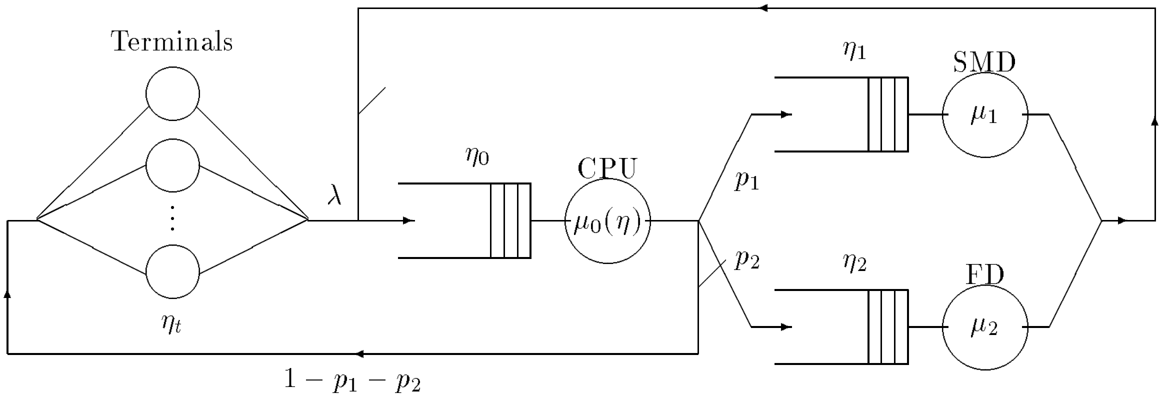

FIG. 1
*An interactive computer system*

Given the necessary parameters, MARCA generates the transition probability matrix corresponding to the specific queueing system under consideration. Experiments with different combinations of parameters are carried out because the effect of making the transition probability matrix larger, and closer to the identity matrix so that it is almost decomposable is to be observed. The generated matrix has a block tridiagonal structure with a small percentage of nonzero elements. It must be remarked that this type of matrix is frequently encountered in queueing network analysis.

**(a)** For the first example, the following parameters are chosen:

$$\begin{aligned}
\eta_t + \eta &= 3 \\
\lambda &= (10^{-4})\eta_t \\
p_1\mu_0(\eta) &= 100(\eta/128)^{1.5} \\
p_2\mu_0(\eta) &= 0.05 \\
1 - p_1 - p_2 &= 0.002 \\
\mu_1 &= 0.2 \\
\mu_2 &= 1/30
\end{aligned}$$

Table 4

*Sizes of the aggregates for Problem 2(a):* $\gamma = 10^{-3}$.

| | | | |
|---|---|---|---|
| 10 | 6 | 3 | 1 |

Table 5

*Results for Problem 2(a):* $n = 20, n_z = 80$, $\gamma = 10^{-3}, N = 4, nzb = 10$.

| $M_{agg}$ | $M_{disagg}$ | *Iter* | $T_{total}$ | $Err_{res}$ | $Err_{rel}$ |
|---|---|---|---|---|---|
| $GE_1$ | $GE_1$ | 5 | 0.10 | $0.468E - 15$ | $0.381E - 12$ |
| $GE_2$ | $GE_2$ | 5 | 0.10 | $0.483E - 15$ | $0.391E - 12$ |
| $GTH_2$ | $GTH_2$ | 5 | 0.10 | $0.447E - 15$ | $0.404E - 12$ |
| $GTH_1$ | $GTH_1$ | 5 | 0.10 | $0.448E - 15$ | $0.404E - 12$ |

**(b)** The parameters in this example are the same as those in 2(a) except

$$\mu_2 = (10^{-10})/30.$$

Sizes of the aggregates for Problem 2(b) for $\gamma = 10^{-11}$ are the same as for Problem 2(a).

Table 6

*Results for Problem 2(b):* $n = 20, n_z = 80$, $\gamma = 10^{-11}, N = 4, nzb = 10$.

| $M_{agg}$ | $M_{disagg}$ | *Iter* | $T_{total}$ | $Err_{res}$ | $Err_{rel}$ |
|---|---|---|---|---|---|
| $GE_1$ | $GE_1$ | 1 | 0.07 | $0.974E - 17$ | $0.345E - 14$ |
| $GE_2$ | $GE_2$ | 1 | 0.07 | $0.974E - 17$ | $0.345E - 14$ |
| $GTH_2$ | $GTH_2$ | 1 | 0.07 | $0.999E - 20$ | $0.421E - 16$ |
| $GTH_1$ | $GTH_1$ | 1 | 0.07 | $0.999E - 20$ | $0.421E - 16$ |

**(c)** The parameters in this example are the same as those in 2(a) except

$$\mu_2 = (10^{-14})/30.$$

Sizes of the aggregates for Problem 2(c) for $\gamma = 10^{-15}$ are the same as for Problem 2(a).

Table 7

*Results for Problem 2(c):* $n = 20, n_z = 80$, $\gamma = 10^{-15}, N = 4, nzb = 10$.

| $M_{agg}$ | $M_{disagg}$ | *Iter* | $T_{total}$ | $Err_{res}$ | $Err_{rel}$ |
|---|---|---|---|---|---|
| $GE_1$ | $GE_1$ | 1 | 0.02 | $0.551E - 16$ | $0.204E - 13$ |
| $GE_2$ | $GE_2$ | 1 | 0.02 | $0.551E - 16$ | $0.204E - 13$ |
| $GTH_2$ | $GTH_2$ | 1 | 0.02 | $0.846E - 28$ | $0.354E - 24$ |
| $GTH_1$ | $GTH_1$ | 1 | 0.02 | $0.841E - 28$ | $0.354E - 24$ |

**(d)** The parameters in this example are the same as those in 2(a) except

$$\eta_t + \eta = 10.$$

Table 8
*Sizes of the aggregates for Problem 2(d): $\gamma = 10^{-3}$.*

| | | | | | | |
|---|---|---|---|---|---|---|
| 1 | 3 | 6 | 10 | 15 | 21 | 28 |
| 36 | 45 | 55 | 66 | | | |

Table 9
*Results for Problem 2(d): $n = 286, n_z = 1,606$, $\gamma = 10^{-3}, N = 11, nzb = 31$.*

| $M_{agg}$ | $M_{disagg}$ | *Iter* | $T_{total}$ | $Err_{res}$ | $Err_{rel}$ |
|---|---|---|---|---|---|
| $GE_1$ | $GE_1$ | 8 | 0.47 | $0.519E-16$ | $0.142E-11$ |
| $GE_2$ | $GE_2$ | 8 | 0.64 | $0.969E-16$ | $0.469E-11$ |
| $GTH_2$ | $GTH_2$ | 8 | 1.18 | $0.412E-16$ | $0.233E-12$ |
| $GTH_1$ | $GTH_1$ | 8 | 2.72 | $0.449E-16$ | $0.233E-12$ |

**(e)** The parameters in this example are the same as those in 2(d).

Table 10
*Sizes of the aggregates for Problem 2(e): $\gamma = 10^{-4}$.*

| | | |
|---|---|---|
| 65 | 55 | 165 |

Table 11
*Results for Problem 2(e): $n = 286, n_z = 1,606$, $\gamma = 10^{-4}, N = 3, nzb = 7$.*

| $M_{agg}$ | $M_{disagg}$ | *Iter* | $T_{total}$ | $Err_{res}$ | $Err_{rel}$ |
|---|---|---|---|---|---|
| $GE_1$ | $GE_1$ | 5 | 0.88 | $0.583E-16$ | $0.178E-11$ |
| $GE_2$ | $GE_2$ | 5 | 1.09 | $0.531E-16$ | $0.132E-11$ |
| $GTH_2$ | $GTH_2$ | 5 | 2.89 | $0.398E-16$ | $0.383E-12$ |
| $GTH_1$ | $GTH_1$ | 5 | 13.84 | $0.350E-16$ | $0.383E-12$ |

**(f)** The parameters in this example are the same as those in 2(d) except

$$\mu_2 = (10^{-5})/30.$$

Table 12
*Sizes of the aggregates for Problem 2(f): $\gamma = 10^{-3}$.*

| | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | 2 | 1 | 3 | 2 | 4 | 1 | 3 | 5 | 2 | 4 | 6 | 1 | 3 | 5 | 7 |
| 2 | 4 | 6 | 8 | 1 | 3 | 5 | 7 | 9 | 2 | 4 | 6 | 8 | 10 | | |
| 1 | 3 | 5 | 7 | 9 | 11 | 2 | 4 | 6 | 8 | 10 | 1 | 3 | 5 | 7 | 9 |
| 2 | 4 | 6 | 8 | 1 | 3 | 5 | 7 | 2 | 4 | 6 | 1 | 3 | 5 | 2 | 4 |
| 1 | 3 | 2 | 1 | | | | | | | | | | | | |

TABLE 13
*Results for Problem 2(f):* $n = 286, n_z = 1,606,\ \gamma = 10^{-3}, N = 66, nzb = 196.$

| $M_{agg}$ | $M_{disagg}$ | *Iter* | $T_{total}$ | $Err_{res}$ | $Err_{rel}$ |
|---|---|---|---|---|---|
| $GE_1$ | $GE_1$ | 3 | 0.16 | $0.247E-16$ | $0.469E-13$ |
| $GE_2$ | $GE_2$ | 3 | 0.24 | $0.247E-16$ | $0.469E-13$ |
| $GTH_2$ | $GTH_2$ | 3 | 0.39 | $0.111E-15$ | $0.107E-14$ |
| $GTH_1$ | $GTH_1$ | 3 | 0.49 | $0.111E-15$ | $0.107E-14$ |

**(g)** The parameters in this example are the same as those in 2(a) except

$$\eta_t + \eta = 20 \text{ and } \lambda = (10^{-7})\eta_t.$$

TABLE 14
*Sizes of the aggregates for Problem 2(g):* $\gamma = 10^{-6}$.

| | | | | | | |
|---|---|---|---|---|---|---|
| 1 | 3 | 6 | 10 | 15 | 21 | 28 |
| 36 | 45 | 55 | 66 | 78 | 91 | 105 |
| 120 | 136 | 153 | 171 | 190 | 210 | 231 |

TABLE 15
*Results for Problem 2(g):* $n = 1,771, n_z = 11,011,\ \gamma = 10^{-6}, N = 21, nzb = 61.$

| $M_{agg}$ | $M_{disagg}$ | *Iter* | $T_{total}$ | $Err_{res}$ | $Err_{rel}$ |
|---|---|---|---|---|---|
| $GE_1$ | $GE_1$ | 3 | 3.79 | $0.234E-16$ | $0.263E-12$ |
| $GE_2$ | $GE_2$ | 3 | 4.11 | $0.234E-16$ | $0.263E-12$ |
| $GTH_2$ | $GTH_2$ | 3 | 15.12 | $0.514E-18$ | $0.605E-14$ |
| $GTH_1$ | $GTH_1$ | 3 | 82.47 | $0.514E-18$ | $0.605E-14$ |

**(h)** The parameters in this example are the same as those in 2(a) except

$$\eta_t + \eta = 20,\ \ \mu_1 = 0.2(10^{-10}),\ \text{ and }\ \mu_2 = (10^{-10})/30.$$

Sizes of the aggregates for Problem 2(h) for $\gamma = 10^{-12}$ are the same as for Problem 2(g).

TABLE 16
*Results for Problem 2(h):* $n = 1,771, n_z = 11,011,\ \gamma = 10^{-12}, N = 21, nzb = 61.$

| $M_{agg}$ | $M_{disagg}$ | *Iter* | $T_{total}$ | $Err_{res}$ | $Err_{rel}$ |
|---|---|---|---|---|---|
| $GE_1$ | $GE_1$ | 3 | 2.23 | $0.390E-16$ | $0.466E-03$ |
| $GE_2$ | $GE_2$ | 3 | 2.88 | $0.390E-16$ | $0.466E-03$ |
| $GTH_2$ | $GTH_2$ | 3 | 9.12 | $0.399E-16$ | $0.583E-15$ |
| $GTH_1$ | $GTH_1$ | 3 | 26.89 | $0.399E-16$ | $0.584E-15$ |

In all problems considered, both IAD with GE and IAD with GTH converged (see *Iter* in Table 1) with a residual of order machine epsilon regardless of the implementation scheme chosen. For each problem, the number of iterations taken by both methods is the same. However, the relative error in IAD with GE is much larger than that of IAD with GTH in the first small problem (see Table 3) where IAD with GTH has a relative error almost of order machine epsilon. It is presumed this is due to the smaller size of the systems solved in this problem. The first test problem has a small coefficient matrix giving rise to a coupling matrix of size $3 \times 3$. Similarly, the largest system solved in the disaggregation step of the IAD algorithm is of size $3 \times 3$. Consequently, one expects to lose a relative accuracy of roughly 2 digits in this problem when IAD is used with GTH. The results confirm that IAD with GTH performs even better than the expectations, and only a single digit of accuracy is lost. However, as in the second problem, there are cases where the GTH algorithm does not achieve its worst case bound due to statistical effects in rounding error accumulation although the matrices solved are considerably larger (see Tables 4, 8, 10, 12, and 14). Moreover, just as indicated in [10], it is possible to improve the entrywise relative error in the GTH algorithm even further by forming the pivot element in higher precision (quadruple–precision for the given system parameters).

As expected, the number of iterations taken to achieve the prespecified tolerance criterion decreases as the decomposability parameter becomes smaller for a given problem. Among the problems considered, when the decomposability parameter is greater than or equal to $10^{-6}$, IAD with GE provided results in which 3 to 5 digits of accuracy are lost (see Tables 3, 5, 9, 11, 13, and 15). An observation regarding the results of the second test problem is the difference between $T_{total}$'s for implementation Schemes 1 and 2 (see Tables 9, 11, 13, 15, and 16). First, although it is not the case for the second problem, the nonzero structure of a nonsymmetric matrix changes if the matrix is transposed, thus totally affecting the reduction process. Second, both methods in Scheme 2 store the multipliers, and thus they spend extra time. Finally, remember that Scheme 1 implementation of GTH is accomplished by shifting data around in memory, a very time consuming process.

The results of the modified scheme (suggested by G. W. Stewart) are quite competitive with those of IAD with GTH for the first small problem (see Table 17). However, there are examples (Test problem 2, parts (b), (c), (e), (f), (g)) with varying degrees of decomposability for which the modified scheme does not provide a relative error that is competitive with the relative error in IAD with GTH (see Tables 6, 7, 11, 13, and 15). However, it is never larger than the relative error in IAD with GE (see Table 17). The timing of the modified scheme is almost as good as that of IAD with GE even for the larger test cases.

Table 17
*Results for modified scheme (transposed version).*

| *Problem* | *Iter* | $T_{total}$ | $Err_{res}$ | $Err_{rel}$ |
|---|---|---|---|---|
| 1 | 4 | 0.04 | $0.621E-16$ | $0.423E-15$ |
| 2(a) | 5 | 0.10 | $0.458E-15$ | $0.381E-12$ |
| 2(b) | 1 | 0.07 | $0.974E-17$ | $0.345E-14$ |
| 2(c) | 1 | 0.02 | $0.551E-16$ | $0.204E-13$ |
| 2(d) | 8 | 0.56 | $0.443E-16$ | $0.245E-12$ |
| 2(e) | 5 | 1.03 | $0.385E-16$ | $0.109E-11$ |
| 2(f) | 3 | 0.28 | $0.247E-16$ | $0.469E-13$ |
| 2(g) | 3 | 4.35 | $0.234E-16$ | $0.263E-12$ |
| 2(h) | 3 | 2.71 | $0.399E-16$ | $0.273E-15$ |

# 6 Conclusion

In this paper, the computation of the stationary probability vectors of ill–conditioned NCD Markov chains is considered. The GTH method, which avoids subtractions, is a much more stable version of GE. For that reason, it is a good candidate to be used in the two–level IAD algorithm. Experiment on several problems are carried out, applying this idea versus GE in the IAD technique. The GTH approach to calculating the pivot element by taking the negated sum of the off–diagonal elements in the unreduced part of the pivot column proves to be valuable for singular M–matrices with 0 column sums, and it is shown to be quite effective on the problems of interest.

The GTH idea is employed to solve the linear systems of equations formed in both the aggregation and disaggregation steps of the IAD algorithm. For the first small problem considered, we observed that the relative error in IAD with GE is much larger than that of IAD with GTH, whose relative error is fixed in the order of machine epsilon. In all cases, the size of the systems solved provides an estimation for the worst case bound on the relative error for IAD with GTH. Additionally, just as explained in §4, it was not surprising to see IAD with GTH take on the order of 10 times as much time as IAD with GE for the larger chains in the second test problem. A modified scheme suggested by G. W. Stewart, which essentially performs diagonal correction on pivots only when there is a suspected loss in significance, seems to work very well in the IAD algorithm for small NCD Markov chains. The scheme possesses the good relative accuracy property of GTH and the convenient implementation of GE. However, for larger matrices, it is frequently no better than GE (while never being worse than GE) in terms of relative accuracy. It is also verified for IAD with GE that a significant difference between the relative error and the residual error is a clear indication of an ill–conditioned problem.

In conclusion, ordinary GE should definitely be avoided in both steps of the iterative IAD algorithm when solving NCD chains with a degree of coupling less than machine epsilon. However, if an approximation of the stationary vector of a large NCD Markov chain is sought in a short time, IAD with GE may be used. On the contrary, if relative error in the stationary vector of the NCD chain is deemed as of utmost importance, then IAD with GTH has to be recommended. However, we do recommend IAD with GE when the decomposability parameter for a given problem is greater than or equal to $10^{-6}$. Only 3 to 5 digits of accuracy were lost in such problems considered. A compromise between GE and GTH seems to be the modified scheme suggested by G. W. Stewart. Examination of several sparse storage formats for both GTH and GE has indicated one disadvantage of the GTH method. The time to execute the IAD algorithm on large irreducible NCD Markov chains tends to be longer when the GTH method is used in the aggregation and disaggregation steps of the iterative solver. Memory requirement of the GTH algorithm is generally slightly higher than that of GE; nevertheless this mostly depends on the sparse storage format chosen. One last remark would be to direct the attention to the possibility of exploiting the inherent parallelism in the formation of the coupling matrix. Similarly, parallel implementation of the solution of $N$ nonsingular systems in the disaggregation step needs to be investigated.

# References

[1] A. Berman and R. J. Plemmons, *Nonnegative Matrices in the Mathematical Sciences*, Academic Press, New York, 1979.

[2] W. L. Cao and W. J. Stewart, *Iterative aggregation/disaggregation techniques for nearly uncoupled Markov chains*, J. Assoc. Comput. Mach., 32 (1985), pp. 702–719.

[3] P.–J. Courtois, *Decomposability: Queueing and Computer System Applications*, Academic Press, New York, 1977.

[4] W. K. Grassmann, M. I. Taksar and D. P. Heyman, *Regenerative analysis and steady state distributions for Markov chains*, Oper. Res., 33 (1985), pp. 1107–1116.

[5] W. J. Harrod and R. J. Plemmons, *Comparison of some direct methods for computing the stationary distributions of Markov chains*, SIAM J. Sci. Statist. Comput., 5 (1984), pp. 453–469.

[6] D. P. Heyman, *Further comparisons of direct methods for computing stationary distributions of Markov chains*, SIAM J. Sci. Statist. Comput., 8 (1987), pp. 226–232.

[7] J. R. Koury, D. F. McAllister and W. J. Stewart, *Iterative methods for computing stationary distributions of nearly completely decomposable Markov chains*, SIAM J. Alg. Discrete Meth., 5 (1984), pp. 164–186.

[8] C. D. Meyer, *Stochastic complementation, uncoupling Markov chains, and the theory of nearly reducible systems*, SIAM Rev., 31 (1989), pp. 240–272.

[9] ———, *Sensitivity of the stationary distribution of a Markov chain*, SIAM J. Matrix Anal. Appl., 15 (1994), pp. 715–728.

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

[11] C. C. Paige, P. H. Styan and P. G. Wachter, *Computation of the stationary distribution of a Markov chain*, J. Statist. Comput. Simulation, 4 (1975), pp. 173–186.

[12] B. Philippe, Y. Saad and W. J. Stewart, *Numerical methods in Markov chain modelling*, Oper. Res., 40 (1992), pp. 1156–1179.

[13] S. Pissanetzky, *Sparse Matrix Technology*, Academic Press, London, 1984.

[14] P. J. Schweitzer, *A survey of aggregation–disaggregation in large Markov chains*, in Numerical Solution of Markov Chains, W. J. Stewart, ed., Marcel Dekker, Inc., New York, 1991, pp. 63–88.

[15] H. Simon and A. Ando, *Aggregation of variables in dynamic systems*, Econometrica, 29 (1961), pp. 111–138.

[16] G. W. Stewart and G. Zhang, *On a direct method for the solution of nearly uncoupled Markov chains*, Numer. Math., 59 (1991), pp. 1–11.

[17] G. W. Stewart, W. J. Stewart and D. F. McAllister, *A two–stage iteration for solving nearly completely decomposable Markov chains*, in The IMA Volumes in Mathematics and its Applications 60: Recent Advances in Iterative Methods, G. H. Golub, A. Greenbaum, and M. Luskin, eds., Springer–Verlag, New York, 1994, pp. 201–216.

[18] G. W. Stewart, personal communication, 1994.

[19] W. J. Stewart, *A comparison of numerical techniques in Markov modelling*, Comm. ACM, 21 (1978), pp. 144–152.

[20] ———, *MARCA: Markov Chain Analyzer, A software package for Markov modeling*, in Numerical Solution of Markov Chains, W. J. Stewart, ed., Marcel Dekker, Inc., New York, 1991, pp. 37–61.

[21] W. J. Stewart and W. Wu, *Numerical experiments with iteration and aggregation for Markov chains*, ORSA J. Comput., 4 (1992), pp. 336–350.

[22] W. J. Stewart, *Introduction to the Numerical Solution of Markov Chains*, Princeton University Press, Princeton, NJ, 1994.

[23] Y. Takahashi, *A lumping method for the numerical calculation of stationary distributions of Markov chains*, Research Report B–18, Dept. of Information Sciences, Tokyo Institute of Technology, 1975.