INFORMS Journal on Computing
Vol. 16, No. 3, Summer 2004, pp. 300–315
ISSN 0899-1499 | EISSN 1526-5528 | 04 | 1603 | 0300

DOI 10.1287/ijoc.1030.0041

Testing the Nearest Kronecker Product Preconditioner on Markov Chains and Stochastic Automata Networks

Amy N. Langville
Operations Research Program, North Carolina State University, Raleigh, North Carolina 27695-7913, USA,
anlangvi@unity.ncsu.edu

William J. Stewart
Department of Computer Science, North Carolina State University, Raleigh, North Carolina 27695-8206, USA,
billy@csc.ncsu.edu

This paper is the experimental follow-up to Langville and Stewart (2002), where the theoretical background for the nearest Kronecker product (NKP) preconditioner was developed. Here we test the NKP preconditioner on both Markov chains (MCs) and stochastic automata networks (SANs). We conclude that the NKP preconditioner is not appropriate for general MCs, but is very effective for a MC stored as a SAN.

Key words: probability; Markov processes; queues; Markovian; algorithms
History: Accepted by Edward P. C. Kao; received May 2002; revised April 2003; accepted May 2003.

1. Introduction

Markov chains (MCs) grow exponentially with the number of state variables and thus, in practice, Markov-chain modelers often encounter very large matrices, which makes the analysis difficult. Many large MCs can be effectively defined and analyzed using the newer model of stochastic automata networks (SANs). SANs, which were first proposed by Plateau (1985), have been used since to represent the generator matrix Q compactly. Plateau and her coworkers have shown that a SAN can be represented in compact form as a sum of Kronecker products, known as the *SAN descriptor* (Plateau 1985, Plateau and Fourneau 1991, Stewart 1994).

In this paper, we are concerned with computing the stationary solution of a MC that has been compactly represented and stored as a SAN. Most of the traditional stationary-solution techniques for MCs can be applied to SANs. However, direct methods for solving linear systems, such as those based on LU decompositions, are not immediately amenable to SANs because the SAN's compact descriptor representation of the generator matrix precludes easy access to the L and U factors. Furthermore, SANs are used as a compact, alternative representation for very large Markov models and the size of such models makes direct methods impractical (Stewart 1994).

In constrast, iterative and projection methods for solving linear systems have been studied extensively. Many methods in this class have been successfully applied to SANs. Originally, it was thought that the classical iterative methods of Gauss-Seidel, and successive overrelaxation (SOR) as well as the aggregation/disaggregation methods, could not be easily applied to SANs (Stewart 1994). However, recently, Dayar and his colleagues have done work on the Gauss-Seidel method for SANs (Dayar 1998, Uysal and Dayar 1998), while Buchholz and his coworkers have developed special multiplication algorithms making Gauss-Seidel and Jacobi applicable to SANs (Buchholz et al. 2000). In a more straightforward manner, iterative methods, whose only interaction with the coefficient matrix involves computing a vector-matrix product, have been employed to compute the stationary probabilities for SANs. Since SANs use descriptors rather than matrices to represent the system and its transitions, such methods incorporate the efficient vector-Kronecker product multiplication algorithm described in Fernandes et al. (1998). These methods with straightforward SAN application include the power method and projection methods such as generalized minimal residual (GMRES) and Arnoldi. GMRES and Arnoldi both use long recurrences, requiring that a large number of vectors be stored, thereby increasing the work at each iteration. Projection methods with shorter recurrences, namely biconjugate gradient stabilized (BiCGSTAB), conjugate gradient squared (CGS), and transpose-free quasi-minimal residual (TFQMR) have also been used to find the stationary vector of a SAN (Buchholz 1999).

Due to the close relationship between SANs and MCs, preconditioning techniques that accelerate the convergence of iterative methods applied to MCs have been employed to accelerate convergence in SANs. Philippe et al. (1992), Saad (1995), and Stewart (1994) review preconditioners for MCs, of which the incomplete LU factorization preconditioners have enjoyed much success. Unfortunately, these preconditioners cannot be readily used for SANs for the same reasons that direct methods based on the L and U factors cannot be used (Stewart et al. 1995). A few other preconditioners tailored to SANs have been proposed (Stewart et al. 1995, Buchholz 1999). Yet these SAN preconditioners have proven to be of little help in reducing the work and time involved in reaching the stationary solution. Much more work remains to be done to develop efficient preconditioners, especially for very large systems modeled as SANs.

Our aim is to address this need by examining the plausibility of one particular approximate preconditioner, the nearest Kronecker product (NKP) approximate preconditioner. This preconditioner can then be used with any of the iterative or projection methods in order to compute stationary probabilities.

2. Iterative Methods

In this paper we restrict ourselves to the problem of finding the stationary solution vector of a large MC represented as a SAN. The results of this paper can also be applied to certain methods for computing transient solutions.

We present the iterative step of the most basic iterative method, the *power method*, applied to SANs. The generator matrix Q is replaced with the SAN descriptor, $\sum_{j=1}^{T} \bigotimes_{i=1}^{N} Q_j^{(i)}$, where $T = 2E + N$, E is the number of synchronizing events, and N is the number of automata. For ergodic MCs, the power method converges to the dominant eigenvector (the stationary solution vector π) of the transition probability matrix P, where $P = I + \Delta t Q$ and $\Delta t = 1/\max|q_{ii}|$. Then in the SAN formalism,

$$P = I + \Delta t Q = \bigotimes_{i=1}^{N} I_{n_i} + \sum_{j=1}^{T} \Delta t \bigotimes_{i=1}^{N} Q_j^{(i)},$$

and the power method for SANs becomes

$$x^{(k+1)} = x^{(k)}(I + \Delta t Q) = x^{(k)} + \Delta t\, x^{(k)} \left(\sum_{j=1}^{T} \bigotimes_{i=1}^{N} Q_j^{(i)} \right).$$

The power method is the simplest iterative method for finding the stationary solution vector π. The power method was the first method applied to SANs to find π. Despite its slow speed, it still serves as the baseline for comparing various new numerical techniques (Plateau et al. 2001, Stewart et al. 1995, Buchholz 1999). Another class of iterative methods is that of projection methods. These methods approximate an exact solution (in our case, the stationary solution) by building better and better approximations, which are taken from small-dimension subspaces. Some popular projection methods are Arnoldi, GMRES, CGS, BiCGSTAB, and quasi-minimal residual (QMR). Such projection methods have been applied to SANs (Buchholz 1999, Plateau 1990, Stewart et al. 1995). In §6, we test our NKP preconditioner using three popular SAN stationary techniques: the baseline power method, GMRES, and BiCGSTAB. All three require a vector-descriptor multiplication in place of a vector-matrix multiplication and thus are easily implementable with the vector-descriptor multiplication algorithm of Fernandes et al. (1998).

3. Preconditioning

It is well known that the iterative methods discussed above perform better when preconditioners are used. Because the convergence of an iterative method depends on the eigenvalues of the system, the goal of preconditioning is to modify the eigenvalue distribution of the iteration matrix so that convergence is improved while the solution remains unchanged.

In general, for the linear system $Ax = b$, we introduce the preconditioning matrix M and solve $MAx = Mb$. We hope that M is a good approximation of A^{-1} and thus convergence will be rapid. For the Markovian case, where Q is singular, we choose M as an approximation to a generalized inverse of Q (Benzi and Tuma 2002). Additionally, for the case of a MC stored as a SAN, we desire a suitable preconditioner M that exploits the SAN structure.

As mentioned in the introduction, a popular set of preconditioners, ILU preconditioners (Stewart 1994), have largely been dismissed from consideration as SAN preconditioners because they require an incomplete LU factorization of the transition matrix. SANs store the transition matrix as a sum of Kronecker products. Hence, an LU factorization of a SAN descriptor is not easily available. Another SAN preconditioner, based on the Neumann series, was introduced by Stewart et al. (1995). Unfortunately, this Neumann SAN preconditioner is quite expensive and tests of this preconditioner on a queueing-network model report that no real benefit was derived from this preconditioner when compared to the unpreconditioned methods (Stewart et al. 1995). Another possible SAN preconditioner, explored by Buchholz (1999), constructs a preconditioner from the inverses of the individual automata matrices. Unfortunately, in only some cases did this preconditioner reduce the computation time required to find the stationary solution.

Other preconditioners for SANs have recently been proposed by Plateau et al. (2001). One preconditioner is based on the additive Schwartz method, while another preconditioner is based on themultiplicative

Schwartz method. Numerical experiments showed that the additive and multiplicative preconditioners were unsuccessful. They did not beat the unpreconditioned methods and required more computation time. A third preconditioner tested was the diagonal preconditioner. This elementary preconditioner provided interesting results. Numerical experimentation showed that, in some cases, simple diagonal preconditioning reduced the number of iterations without increasing the computation time per iteration.

In §6, we use the three popular SAN preconditioners, Neumann, individual inverse, and diagonal, to compare against our new NKP SAN preconditioner.

3.1. The NKP Preconditioner

In Langville and Stewart (2002), we discovered a theoretically sound approximate inverse preconditioner for the SAN problem—the NKP preconditioner. Given the SAN descriptor, $\sum_{j=1}^{T} \bigotimes_{i=1}^{N} Q_j^{(i)}$, we detail the steps for finding small matrices $A_1, A_2, \ldots, A_N$ such that $A_1 \otimes A_2 \otimes \cdots \otimes A_N$ approximates the SAN descriptor. Then the matrix $M = A_1^{-1} \otimes A_2^{-1} \otimes \cdots \otimes A_N^{-1}$ becomes the preconditioning matrix. The advantage of the Kronecker approximation for SANs is that M need never be formed; instead, only A_1^{-1}, $A_2^{-1}, \ldots, A_N^{-1}$ need to be stored and used in the vector-Kronecker product multiplication of the iterative methods. We note that for our case of MCs, we want to approximate the group inverse $Q^{\#}$ rather than Q^{-1}. However, the algorithm for finding A_1, A_2, …, A_N almost always results in nonsingular matrices A_1, A_2, …, A_N. Thus, we must use the standard inverses, $A_1^{-1}, A_2^{-1}, \ldots, A_N^{-1}$, to form the preconditioner. In effect, we are using the ideal preconditioner $M = A_1^{-1} \otimes A_2^{-1} \otimes \cdots \otimes A_N^{-1}$ for a nearby system whose coefficient matrix $\hat{Q}$ is almost Q. The remainder of this paper contains test examples to determine if this theoretically sound NKP preconditioner is practically sound. The testing section is divided: First, the NKP preconditioner is tested on general MCs (those whose infinitesimal generator is stored in some compact form, but not the SAN descriptor form), and then the NKP preconditioner is tested on SANs.

4. Review of Main Theoretical Results from Langville and Stewart (2002)

In Langville and Stewart (2002), we explain how computation of the NKP preconditioner differs depending on the representation of the infinitesimal generator matrix. When Q is stored in a usual compact matrix representation like the Harwell-Boeing format, we call the stationary analysis problem ($\pi Q = 0$) the *MC problem*. When Q is stored as a SAN descriptor, we call the stationary problem ($\pi \sum_{j=1}^{T} \bigotimes_{i=1}^{N} Q_j^{(i)} = 0$) the *SAN problem*. Our goal is to test the NKP preconditioner on both problems. Before proceeding to the testing sections, (§§5 and 6), we recall the main findings from our derivation of the NKP preconditioner in Langville and Stewart (2002). The relevant findings for applying the NKP preconditioner to a general MC are:

$$\|Q - A_1 \otimes A_2\|_F^2 = \|\tilde{Q} - a_1 a_2^T\|_F^2,$$

where a_1 is the vectorized version of matrix A_1 formed by stacking the columns of A_1, denoted $\text{vec}(A_1)$, $a_2 = \text{vec}(A_2)$, $\tilde{Q}$ is the rearrangement of the infinitesimal generator Q detailed in Pitsianis and Van Loan (1993), and $\|\cdot\|_F$ is the Frobenius norm. The NKP of Q, $A_1 \otimes A_2$, is neither unique nor exact. By design only two NKP matrices, A_1 and A_2, can be found for a general MC since the SVD with its dual components, lefthand and righthand singular vectors, underlies the method. The SVD of $\tilde{Q}$ gives the nearest rank-1 matrix $a_1 a_2^T$ to $\tilde{Q}$. Thus, the optimal $a_1 = \sigma_1 U_1$ and the optimal $a_2 = V_1$, where σ_1 is the largest singular value of $\tilde{Q}$ and U_1, V_1 are the corresponding left and right singular vectors. In fact, $\tilde{Q}$ never needs to be formed or stored, only a $\tilde{Q}$-vector multiplication algorithm is needed (Pitsianis and Van Loan 1993). An iterative SVD algorithm is applied to find σ_1, U_1, and V_1 quickly. A reversal of the vectorizing operation applied to vectors a_1 and a_2 gives the optimal NKP matrices A_1 and A_2. We form $M = A_1^{-1} \otimes A_2^{-1}$ as the NKP preconditioner for the MC stationary system. We note that since the MC does not have an inherent Kronecker structure, the maximum number of NKP matrices that can be obtained is two. Contrast this with the SAN case, where N NKP matrices are formed. The results of the NKP preconditioner applied to MCs are presented in §5.

We now turn to SANs and discuss the derivation of the NKP preconditioner for SANs. In order to form the SAN NKP preconditioner $M = A_1^{-1} \otimes A_2^{-1} \otimes \cdots \otimes A_N^{-1}$, we find the NKP matrices $A_1, A_2, \ldots, A_N$ so that $\|Q - A_1 \otimes A_2 \otimes \cdots \otimes A_N\|_F^2$ is minimized. In Langville and Stewart (2002), we prove that

$$\begin{aligned} A_1 &\approx \alpha_1 Q_1^{(1)} + \alpha_2 Q_2^{(1)} + \cdots + \alpha_T Q_T^{(1)}, \\ A_2 &\approx \beta_1 Q_1^{(2)} + \beta_2 Q_2^{(2)} + \cdots + \beta_T Q_T^{(2)}, \\ &\vdots \\ A_N &\approx \eta_1 Q_1^{(N)} + \eta_2 Q_2^{(N)} + \cdots + \eta_T Q_T^{(N)}. \end{aligned}$$

With these statements, we transform the original problem into

$$\begin{aligned} &\|Q - A_1 \otimes A_2 \otimes \cdots \otimes A_N\|_F^2 \\ &\quad \approx \left\| \sum_{j=1}^{T} \bigotimes_{i=1}^{N} Q_j^{(i)} - \left(\sum_{j=1}^{T} \alpha_j Q_j^{(1)} \right) \otimes \left(\sum_{j=1}^{T} \beta_j Q_j^{(2)} \right) \right. \\ &\qquad \left. \otimes \cdots \otimes \left(\sum_{j=1}^{T} \eta_j Q_j^{(N)} \right) \right\|_F^2 \end{aligned}$$

$$= \left[\sum_{i=1}^{T} \sum_{j=1}^{T} \prod_{k=1}^{N} \operatorname{tr}(Q_i^{(k)T} Q_j^{(k)}) \right] - 2\left(\sum_{i=1}^{T} \left[\prod_{k=1}^{N} \left(\sum_{j=1}^{T} \alpha_j^{(k)} \operatorname{tr}(Q_i^{(k)T} Q_j^{(k)}) \right) \right] \right) + \prod_{k=1}^{N} \left[\sum_{i=1}^{T} \sum_{j=1}^{T} \alpha_i^{(k)} \alpha_j^{(k)} \operatorname{tr}(Q_i^{(k)T} Q_j^{(k)}) \right]. \quad (1)$$

In the above equation, $\alpha_i^{(1)} = \alpha_i$, $\alpha_i^{(2)} = \beta_i$, and $\alpha_i^{(N)} = \eta_i$ for ease of notation. Equation (1) says that finding the NKP matrices for a SAN descriptor is approximately equal to finding the $\alpha_1, \alpha_2, \ldots, \alpha_T$, $\beta_1, \beta_2, \ldots, \beta_T, \ldots, \eta_1, \eta_2, \ldots, \eta_T$ (that is, all the $\alpha_i^{(k)}$s) so that the above nonlinear function is minimized. Find the coefficients by any nonlinear minimization method and form the NKP matrices $A_1, A_2, \ldots, A_N$ according to the rules below.

$$\begin{aligned} A_1 &= \alpha_1 Q_1^{(1)} + \alpha_2 Q_2^{(1)} + \cdots + \alpha_T Q_T^{(1)}, \\ A_2 &= \beta_1 Q_1^{(2)} + \beta_2 Q_2^{(2)} + \cdots + \beta_T Q_T^{(2)}, \\ &\vdots \\ A_N &= \eta_1 Q_1^{(N)} + \eta_2 Q_2^{(N)} + \cdots + \eta_T Q_T^{(N)}. \end{aligned}$$

The exciting part of this transformation is that the advantageous property of the SAN is maintained; no larger matrices are formed. All operations are executed on the smaller $Q_j^{(i)}$ matrices. The additional benefit is that the sole operation performed is the trace calculation, a very efficient operation, which requires $O(n)$ time, where n is the order of the matrix. Equation (1) contains NT variables and requires the computation and storage of $NT(T+1)/2$ traces. Clearly, as N, the number of automata, increases or $T = 2E + N$, where E is the number of synchronizing events, grows, this transformation of the original problem becomes impractical.

This drawback relating to the size of the nonlinear optimization problem can be circumvented with the results of Plateau and her coworkers. In practical modeling situations there are typically many automata. However, the number of states in these automata is typically very small; many have only two states. The large number of automata complicates the analysis of the model since the number of embedded loops in numerical algorithms will also be large. Plateau et al. (2001) show how the original N automata can be grouped together so that the number of automata decreases while the size of each automaton increases. Not only does grouping have the effect of speeding up the underlying algorithms, but, in addition, grouping also reduces the number of synchronizing events or functional transitions and thereby reduces the complexity even further. After grouping the automata suitably, N is often less than five. Practically speaking, five grouped automata each of size 100 would enable analysis of huge MCs of size 10^{10}. Thus, it is reasonable to expect that, after grouping techniques are used, the nonlinear optimization problem in (1) contains fewer than 80 unknowns.

Now that we have gleaned the relevant theoretical NKP findings from Langville and Stewart (2002), we can proceed to test this NKP preconditioner on both stationary analysis problems: the MC problem and the SAN problem.

5. Testing the NKP Preconditioner on MCs

The goal of testing the NKP preconditioner on MCs is to determine if this preconditioner is a good, storage-efficient alternative to the long-standing favorite MC preconditioners, the *ILU* preconditioners.

The preconditioned MC problem is

$$\pi(I - MQ) = 0,$$
$$\pi e = 1,$$

where M is the NKP preconditioner and is designed to approximate $Q^{\#}$. Pitsianis and Van Loan (1993) provide a method for finding the NKP to a given matrix. We use this method to find the matrices A_1 and A_2 such that $\|Q - A_1 \otimes A_2\|_F^2$ is minimized. Then $M = A_1^{-1} \otimes A_2^{-1}$ is the NKP preconditioner. The preferred MC preconditioner is $M = A_1^{\#} \otimes A_2^{\#}$, since Q is singular with rank $n-1$. However, because A_1 and A_2 almost always have full rank, A_1^{-1} and A_2^{-1} must be used. Again, we remind the reader that only two NKP matrices, A_1 and A_2, can be found when applying the NKP preconditioner to MCs.

5.1. Computing the NKP for MCs

Follow the steps below to obtain the NKP preconditioner for the MC system.

1. Choose the dimensions of the NKP matrices A_1 and A_2 so that $A_1 \otimes A_2$ has the same dimension as the square matrix Q. Obtain the largest singular triplet of $\tilde{Q}$, where $\tilde{Q}$ represents the rearrangement of Q given in Pitsianis and Van Loan (1993). This rearrangement depends on the dimensions of A_1 and A_2. The largest singular triplet of $\tilde{Q}$ can be obtained without the formation of $\tilde{Q}$; only a vector-$\tilde{Q}$ multiplication algorithm is needed and is detailed in Pitsianis and Van Loan (1993). The cost of one vector-$\tilde{Q}$ multiplication with a dense Q is $O(n^2)$, where n is the size of Q. A sparse Q can reduce this to considerably less than $O(n^2)$ time. The largest singular triplet can be obtained efficiently using the iterative SVD Lanczos process of Golub et al. (1981).

2. Form the A_1 and A_2 matrices according to $\text{vec}(A_1) = \sigma_1 U_1$ and $\text{vec}(A_2) = V_1$, where σ_1, U_1, and V_1 represent the first singular triplet of $\tilde{Q}$. The matrices A_1 and A_2 are then obtained from the corresponding vectors by a reversal of the vectorizing operation.

3. Form and store A_1^{-1} and A_2^{-1}. This takes less than $O(n^{1.5})$ time if A_1 and A_2 are chosen to have dimension approximately equal to $\sqrt{n}$.

4. Take $M = A_1^{-1} \otimes A_2^{-1}$ as the NKP preconditioner. Only A_1^{-1} and A_2^{-1} need to be stored; M need not be formed since efficient vector-Kronecker product multiplication algorithms exist (Fernandes et al. 1998).

Now a suitable iterative or projection method can be applied to the NKP preconditioned MC system. In the worst case, the NKP preconditioner is computed in $O(n^2)$ time. We state for comparison purposes that the computation time for the *ILU* preconditioner is $O(n^3)$. To test the effectiveness of the NKP preconditioner we use the nine MCs introduced below.

- `tcomm`: A telecommunications network modeling telephone customers on a computerized telephone exchange (Stewart 2003); creates a 666×666 matrix.
- `qnatm`: A multi-class, finite-buffer queueing network for ATM (Stewart 2003); creates a 740×740 matrix.
- `ncd`: A closed queueing network with a central server; generates the nearly completely decomposable MC taken from (Stewart 2003); creates a 286×286 matrix.
- `nonncd`: The ncd system above modified so that it is no longer nearly completely decomposable; creates a 286×286 matrix.
- `twoD`: A two-dimensional MC for biological epidemic model (Stewart 2003); creates a 121×121 matrix.
- `sat`: The wireless communication satellite system from Rouskas et al. (2002); creates a 213×213 matrix.
- `mutex`: A resource-sharing model, where N customers share R resources (Stewart 2003); creates a 794×794 matrix.
- `kanban`: A manufacturing scheduling problem (Krieg and Huhn 2002); creates a 336×336 matrix.
- `courtois`: A small courtois matrix taken from Stewart (1994); creates an 8×8 matrix.

5.2. Effectiveness of the NKP Preconditioner for MCs

In this section, we discuss the effectiveness of the NKP preconditioner in terms of storage and number of iterations until convergence.

5.2.1. Storage. One reason why an analyst might want to use the NKP preconditioner rather than the standard *ILU* preconditioners for the MC problem is storage. The matrices A_1 and A_2 can usually be chosen to have dimension approximately equal to $\sqrt{n}$, where n is the order of the MC matrix Q. This choice for the dimensions of A_1 and A_2 gives the best compromise between storage and the quality of the NKP. The inverses of these smaller matrices are computed and stored. In some instances, storing A_1^{-1} and A_2^{-1} provides considerable savings compared with the storage of the *ILU* matrices, which, although sparse, have the same dimension as Q. Table 1 compares the NKP preconditioner and the *ILU* preconditioner in terms of storage. The notation $nnz(Q)$ refers to the number of nonzero elements in the matrix Q.

In general, the NKP preconditioner severely beats the *ILU* preconditioner in terms of storage, requiring storage of far fewer entries. The few exceptions to this rule each have either the A_1 or A_2 with a large order. For example, the `mutex` matrix is order 794. The only factors of 794 are 2 and 397, thus the modeler has no choice for the dimensions of the NKP matrices A_1 and A_2. The matrix A_1 must be 2×2 and A_2 must be 397×397 or vice versa. The NKP matrices will almost always be dense and this explains the 157,613 $(=397 \cdot 397 + 2 \cdot 2)$ entry in Table 1. The order of the matrix Q often has several factors and the modeler has many choices for the dimensions of A_1 and A_2. Our experience shows that choosing the order of these matrices to be as close to $\sqrt{n}$ as possible gives the best compromise between storage and the accuracy of the NKP. (Choosing the matrix A_1 to be very small, say, of order two, and A_2 to have the remaining dimension, $n/2$, generally gives the most accurate NKP. However, this choice for the order of A_1 and A_2 requires, by far, the greatest storage. Choosing the order of A_1 and A_2 to be approximately $\sqrt{n}$ is the most economical storage choice and only slightly sacrifices accuracy.)

Table 1 Comparison of NKP and *ILU* Preconditioner Storage for 9 MC Matrices

Matrix	tcomm	qnatm	ncd	nonncd	twoD	sat3	mutex	kanban	courtois
nnz(Q)	3,091	4,820	1,606	1,606	441	1,389	6,362	1,416	41
% sparse	0.0070	0.0088	0.0196	0.0196	0.0301	0.0306	0.0101	0.0125	0.6406
nnz(NKPM)	1,693	1,769	653	653	242	5,050	157,613	697	20
% sparse	0.0038	0.0032	0.0080	0.0080	0.0165	0.1113	0.2500	0.0062	0.3125
nnz(ILUM)	3,757	5,560	1,892	1,892	562	1,602	7,156	1,752	49
% sparse	0.0085	0.0102	0.0231	0.0231	0.0384	0.0353	0.0114	0.0155	0.7656

One final comment regarding Table 1 concerns the factors of the order of the matrix Q. It may be possible to "pad" the matrix Q so that the order of the padded Q gives more desirable (closer to $\sqrt{n}$) orders for A_1 and A_2. For the `mutex` example, padding Q (order 794) to create a matrix $\bar{Q}$ of order 800 gives the possibility for a 20×20 matrix A_1 and a 40×40 matrix A_2. This requires the storage of only 2,000 nonzeros for A_1 and A_2 and gives 0.003125 sparsity in contrast to the original 794×794 `mutex` with 157,613 nonzeros and 0.2500 sparsity. This padding of only six dummy states restores the NKP's tremendous storage-saving abilities.

Even further storage savings might be desired. One may prefer to use three NKP matrices A_1, A_2, and A_3 such that $\|Q - A_1 \otimes A_2 \otimes A_3\|_F$ is minimized. However, this is not possible for the MC matrix case because the SVD, which forms only two singular factors (lefthand and righthand), underlies the NKP. (See Langville and Stewart (2002) for a review of this fact.) In contrast, for SANs it is possible to find these smaller matrices A_1, A_2, and A_3 due to the inherent structure of the SAN descriptor $\sum_{j=1}^{2E+N} \bigotimes_{i=1}^{N} Q_j^{(i)}$.

5.2.2. Number of Iterations Until Convergence. With the minimal storage requirements of the NKP preconditioner noted, we now consider the number of iterations needed for an NKP preconditioned iterative method to converge. Table 2 contains the results of experimentation of both the NKP and *ILU* preconditioners applied to the nine MC matrices. The convergence criterion that the maximum norm of the residual vector be less than 10^{-8} is applied in all cases. In Table 2 "—" in the table indicates that the method failed to converge to the stationary solution, but it did converge to the eigenvector corresponding to the dominant eigenvalue. All experiments were conducted on a Sun Ultra 10.

Several important observations from Table 2 are emphasized.

• The NKP power method converged in only one case: `mutex`. This poor behavior stems from the eigenvalue distribution of the NKP iteration matrix $I - (A_1^{-1} \otimes A_2^{-1})Q$, which in most examples has a dominant eigenvalue greater than 1 in magnitude. When this occurs, the NKP power method is guaranteed to fail to converge to the stationary solution. This begs the question "Why does the NKP iteration matrix often have eigenvalues greater than 1 in magnitude?"

If the NKP M does not do a good job of approximating $Q^{\#}$, then MQ is not close enough to $I - e\pi$. The ideal preconditioned MC system has a preconditioned matrix of $I - e\pi$ and converges in one iteration. If MQ is close to $I - e\pi$, its eigenvalues are clustered tightly about one, with one zero eigenvalue due to the singularity of Q.

However, when MQ is not close to $I - e\pi$ it has eigenvalues outside $[0, 2]$. For every eigenvalue of MQ outside $[0, 2]$, $I - MQ$, the iteration matrix for the power method, has eigenvalues greater than one in magnitude and thus the preconditioned power method fails.

Consider two examples: `mutex` and `qnatm`. The NKP preconditioner converges for the `mutex` problem because the eigenvalues of the iteration matrix are all less than or equal to one in magnitude. The five largest in magnitude eigenvalues for the `mutex` iteration matrix are [1 0.8197 0.7712 0.7216 0.6769], while those for `qnatm` are [−1.1305 −1.0635 1 0.9535 −0.9274]. A quick look at the eigenvalue plots of MQ (Figures 1 and 2) shows why `mutex` has all eigenvalues less than or equal to one and `qnatm` does not.

All the eigenvalues of `mutex` MQ are between zero and 1.67, with the majority of the eigenvalues clustered around one as expected, since MQ is designed to approximate $I - e\pi$. (Again, a more effective preconditioned system MQ has a much tighter clustering around one.) Thus, the iteration matrix $I - MQ$ will have all its eigenvalues less than one. On the other hand, the `qnatm` MQ has two eigenvalues greater than two and thus the iteration matrix $I - MQ$ has two eigenvalues greater than one in magnitude.

Table 2 Comparison of NKP and *ILU* Preconditioner Number of Iterations for Nine MC Matrices

Method	tcomm	qnatm	ncd	nonncd	twoD	sat3	mutex	kanban	courtois
Size	666	740	286	286	121	213	794	336	8
Power	10,354	703	>50,000	779	17,956	222	434	147	16,606
NKP Power	—	—	—	—	—	—	91	—	
ILU Power	281	112	5,532	89	13	40	23	44	2
GMRES	>10,000	11	>10,000	16	37	5	6	7	29
NKP GMRES	>10,000	10	>10,000	21	62	4	4	10	20
ILU GMRES	>10,000	3	34	3	2	3	2	3	3
BiCGSTAB	514	39	1,155	47	90	27	34	52	8
NKP BiCGSTAB	428	38	1,411	46	110	26	21	49	8
ILU BiCGSTAB	66	13	74	14	15	10	7	11	3

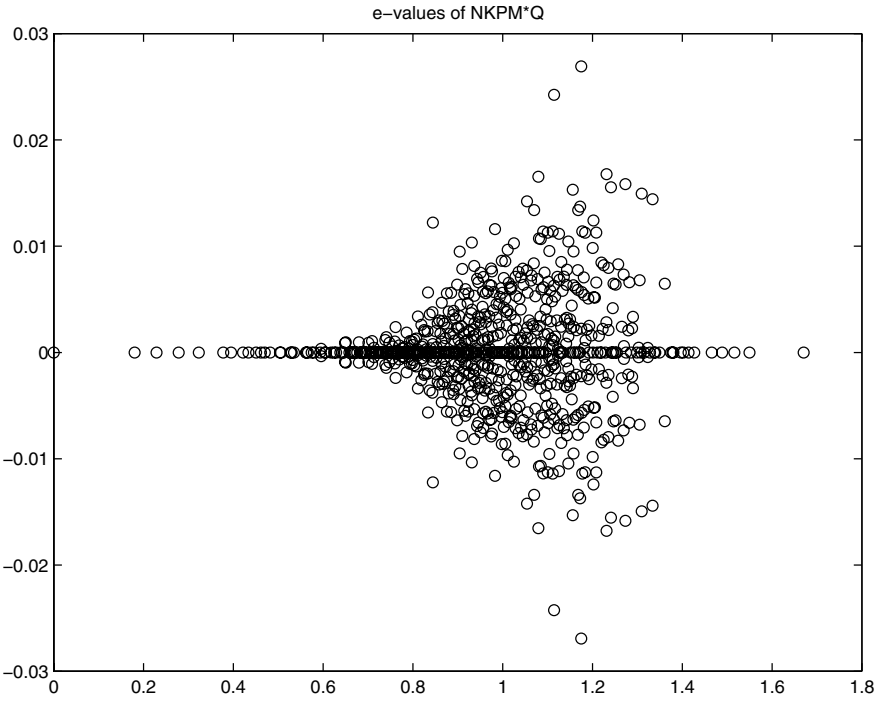

Figure 1 Eigenvalue Plot for NKP Preconditioned *MQ* for `mutex`

• The NKP GMRES method sometimes beats the unpreconditioned GMRES method by a few iterations but never beats the *ILU* GMRES method. There are numerous convergence results for GMRES applied to nonsingular systems (Ipsen and Meyer 1994, Greenbaum 1997, Ipsen 2000, Cao 1997, Toh 1997). However, the majority of these results do not apply to our singular system $\pi Q = 0$. There are a few papers that address GMRES applied to singular systems (Brown and Walker 1997, Freund and Hochbruck

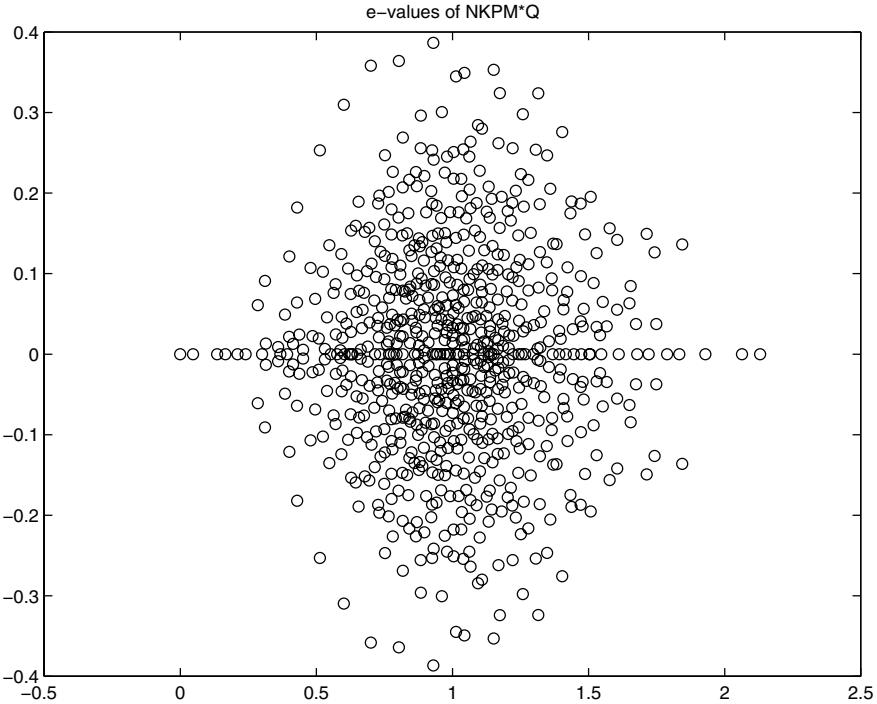

Figure 2 Eigenvalue Plot for NKP Preconditioned *MQ* for `qnatm`

1994), but results relating the convergence behavior of GMRES to the eigenvalue properties of the preconditioned matrix are lacking.

• Sometimes the NKP BiCGSTAB method beats the unpreconditioned BiCGSTAB method by a few iterations. However, the NKP preconditioner still does not compete with the *ILU* preconditioner. Little is known about the convergence properties of BiCGSTAB (Greenbaum 1997) and thus we cannot determine which properties of the NKP preconditioner make it such a poor BiCGSTAB preconditioner.

5.2.3. Why Is the NKP Preconditioner So Bad? Observation of Table 2 lead a researcher to wonder "Why does the NKP preconditioned method not produce greater improvements over the unpreconditioned methods, especially since the NKP is designed to approximate Q?"

Several hypotheses to answer this question were developed in Langville (2002). However, one hypothesis held true. The NKP preconditioner performs perfectly on a coefficient matrix that is deliberately designed as an exact Kronecker product, producing the ideal preconditioned system. Slight perturbations of this Kronecker-structured coefficient matrix make the NKP preconditioner less effective. In fact, as E, a matrix capturing the perturbation of the coefficient matrix away from the Kronecker product, grows, the success of the NKP preconditioner declines rapidly. After many failed hypotheses attempting to justify the inferiority of the NKP preconditioner compared to the *ILU* preconditioner, we arrive at the following conclusion. The reason for the NKP preconditioner's poor performance on MCs is that, in many cases, Q cannot be written as a Kronecker product of two matrices, A_1 and A_2. The NKP found, $A_1 \otimes A_2$, although optimal in the sense of being closest to Q among all Kronecker products, is often just not close enough to Q to serve as the basis for a suitable preconditioner. Forcing the Kronecker structure on a general, unstructured matrix is ineffective.

In those rare cases when Q has Kronecker structure, the NKP preconditioner does a good job, comparable to the *ILU* preconditioner in terms of the number of iterations and superior in terms of storage. Unfortunately, the NKP preconditioner succeeds in few cases. One can predict the relative success of the NKP preconditioner by observing the size of the first singular value of $\tilde{Q}$, σ_1. If σ_1 is very large relative to the sum of all singular values of $\tilde{Q}$, then the NKP $A_1 \otimes A_2$ captures the essence of Q quite well and, consequently, the NKP preconditioner performs well. Table 3 shows the relationship between the size of the perturbation away from an exact Kronecker product and the quality of the NKP. It also shows the relationship between the relative size of the largest singular value and the quality of the NKP. For this table, $Q = B_1 \otimes B_2 + E$, where Q is the sum of an exact Kronecker product $B_1 \otimes B_2$, plus an additive perturbation matrix E, away from this exact Kronecker product.

Table 3 Effect of Perturbations Away from Exact Kronecker Product

$\|E\|_F$	$\|Q - NKP\|_F$	$\sigma_1 / \sum_{i=1}^{n} \sigma_i$	Status
0.1726	0.1571	0.9999	Succeeds
1.7263	1.4077	0.9988	Succeeds
5.1788	4.2848	0.9965	Succeeds
17.2627	14.8018	0.9890	Succeeds
51.9423	47.7126	0.9676	Succeeds
86.3134	74.8579	0.9501	Succeeds
127.2702	102.6748	0.9346	Fails
233.5594	201.3349	0.8521	Fails

In Table 3, the "succeeds" entries refer to the NKP preconditioned method's success over the unpreconditioned methods in terms of number of iterations until convergence. For numerous experiments, the sizes of B_1 and B_2 were varied as well as the structure, but Table 3 reports only the cases in which B_1 and B_2 are both random 10×10 matrices. We also experimented with the perturbation matrix E. In the reported case, E is a 100×100 matrix with tridiagonal structure. Other structures for E were used, but since the tridiagonal structure seemed representative we report only those results. Our observations lead to a general rule of thumb: in most cases, when the ratio of the largest singular value to the sum of all singular values is greater than 0.95, the NKP performs well since the matrix Q is not far from an exact Kronecker product.

The NKP preconditioner's ineffectiveness in reducing the number of iterations until convergence of the iterative methods obviates the need to examine the time needed for its computation.

5.3. Conclusions of the NKP Preconditioner Applied to MCs

As a result of its poor performance, we do not recommend using the NKP preconditioner for a general, unstructured MC. It is not worth the computational time ($O(n^2)$, in the worst case, where n is the order of Q) needed to create it. If, however, the modeler knows or suspects that the Q matrix for the MC has some underlying Kronecker structure, then the NKP preconditioner may be a viable, storage-efficient alternative to the *ILU* preconditioner.

6. Testing the NKP Preconditioner on SANs

The failure of the NKP preconditioner for MCs does not discourage us from trying this preconditioner on SANs. In fact, we are encouraged by the reason for the

NKP preconditioner's failure on MCs. The NKP preconditioner is not appropriate for a MC with no inherent Kronecker structure. SANs have such inherent Kronecker structure and so we predict that the NKP preconditioner might enjoy some success. However, this success may be limited to SANs that have very infrequent interaction among automata.

The NKP preconditioned SAN problem is

$$\pi\left(I - M\left(\sum_{j=1}^{2E+N}\bigotimes_{i=1}^{N} Q_j^{(i)}\right)\right) = 0,$$
$$\pi e = 1$$

where M is the NKP preconditioner and is designed to approximate $Q^{\#}$, the ideal preconditioner. In Langville and Stewart (2002), we thoroughly discuss how to extend Pitsianis and Van Loan's (1993) two-dimensional NKP to an N-dimensional NKP. We find matrices $A_1, A_2, \ldots, A_N$ such that $\| \sum_{j=1}^{2E+N} \bigotimes_{i=1}^{N} Q_j^{(i)} - A_1 \otimes A_2 \otimes \cdots \otimes A_N \|_F$ is minimized. Then, we take $M = A_1^{-1} \otimes A_2^{-1} \otimes \cdots \otimes A_N^{-1}$ as the NKP preconditioner. The preferred preconditioner uses the group inverse, but the nonsingularity of $A_1, A_2, \ldots, A_N$ forces the use of the standard inverse.

In this section, we first outline the computation of the NKP, $A_1 \otimes A_2 \otimes \cdots \otimes A_N$, and then discuss its effectiveness as a SAN preconditioner.

6.1. Computing the Approximate NKP for SANs

Below is an outline for computing the approximate NKP for a SAN. For notational convenience, we let $T = 2E + N$, the total number of Kronecker product terms in the descriptor sum.

1. Form the traces of $(Q_i^{(k)T} Q_j^{(k)})$ for all $i, j = 1, \ldots, T$, $k = 1, \ldots, N$. In total $NT(T+1)/2$ traces must be computed and stored. Each trace operation requires $O(n_i n_j)$ time, where n_i is the order of $Q_i^{(k)}$ and n_j is the order of $Q_j^{(k)}$. Unless these trace operations are executed in parallel, this step requires $O(NT(T+1)n_i n_j/2)$ time.

2. Solve the minimization problem of (1) for NT variables.

3. Form the small approximate NKP matrices: $A_1 = \alpha_1 Q_1^{(1)} + \alpha_2 Q_2^{(1)} + \cdots + \alpha_T Q_T^{(1)}$, $A_2 = \beta_1 Q_1^{(2)} + \beta_2 Q_2^{(2)} + \cdots + \beta_T Q_T^{(2)}$, and $A_N = \eta_1 Q_1^{(N)} + \eta_2 Q_2^{(N)} + \cdots + \eta_T Q_T^{(N)}$. The NKP matrix A_i is restricted to have the same order as $Q_j^{(i)}$. This differs from the two-dimensional NKP case where the modeler has some choices for the size of the NKP matrices A_1 and A_2.

4. Form and store $A_1^{-1}, A_2^{-1}, \ldots, A_N^{-1}$.

5. Take $M = A_1^{-1} \otimes A_2^{-1} \otimes \cdots \otimes A_N^{-1}$ as the NKP preconditioner. Only the small matrices need to be stored; M need not be formed since efficient vector-Kronecker product algorithms exist (Fernandes et al. 1998).

Now a favorite iterative or projection method can be applied to the NKP preconditioned SAN system. The computational effort for obtaining the NKP preconditioner for a SAN is dominated by Step 2, solving the minimization of (1). Limiting the number of unknowns in the minimization problem by grouping, thus reducing N and T (Plateau et al. 2001), is clearly advantageous.

We introduce the seven example SANs used in Langville (2002) to test the effectiveness of the SAN NKP preconditioner.

- `san2`: A SAN derived from a two-automata queueing network (Stewart 1994, Langville 2002). Changing the size of the automata gives SANs of varying global order. We test SANs of order 100, 400, and 1,024. The number of automata, N, is two, the number of terms in the SAN descriptor sum, T, is four, and the number of variables, NT, in the nonlinear minimization problem of (1) is eight.
- `san3A`: A SAN with three automata and two synchronizing events. SANs of various sizes tested were of order 80, 441, and 1000 (Langville 2002). $N = 3$, $T = 7$, and $NT = 21$.
- `san3B`: A SAN with three automata and two synchronizing events. SANs of various sizes tested were of order 80, 441, and 1,000 (Langville 2002). $N = 3$, $T = 7$, and $NT = 21$. Example `san3B` is slightly more complex than example `san3A`.
- `san4A`: A SAN with four automata and two synchronizing events. SANs of various sizes tested were of order 108, 448, and 1,000 (Langville 2002). $N = 4$, $T = 8$, and $NT = 32$.
- `san4B`: A SAN with four automata and three synchronizing events. SANs of various sizes tested were of order 108, 448, and 1,000 (Langville 2002). $N = 4$, $T = 10$, and $NT = 40$. Example `san4B` is more complex than example `san4A`.
- `san5A`: A SAN with five automata and two synchronizing events. SANs of various sizes tested were of order 108, 432, and 1,024 (Langville 2002). $N = 5$, $T = 9$, and $NT = 45$.
- `san5B`: A SAN with five automata and four synchronizing events. SANs of various sizes tested were of order 108, 432, and 1,024 (Langville 2002). $N = 5$, $T = 13$, and $NT = 65$. Example `san5B` is the most complex example tested.

6.2. Effectiveness of the NKP Preconditioner for SANs

We consider three aspects regarding the effectiveness of the NKP preconditioner for SANs: storage, number of iterations until convergence to the stationary solution, and computation time.

6.2.1. Storage. We compare the various SAN preconditioners in terms of the number of nonzeros

Table 4 Comparison of SAN Preconditioner Storage for Seven SANs

SAN	san2	san3A	san3B	san4A	san4B	san5A	san5B
nnz(Q)	4,961	5,500	7,120	7,308	7,308	7,168	6,400
nnz(SAN)	218	110	110	74	84	54	62
nnz(NeumannM)	0	0	0	0	0	0	0
nnz(Inv.Ind.M)	1,552	255	255	118	118	68	62
nnz(DiagM)	0	0	0	0	0	0	0
nnz(NKPM)	251	84	84	61	61	47	50
nnz(ILU0M)	5,985	6,491	8,120	8,388	8,388	8,192	7,424
nnz(ILUTHM)	6,155	5,743	6,747	8,392	8,399	7,896	7,331

stored. For example, the simple diagonal preconditioner requires no additional storage, hence $nnz(DiagM) = 0$ for all examples. We include the number of nonzeros needed by the *ILU* preconditioners ($nnz(ILU0M)$ and $nnz(ILUTHM)$ for comparison purposes only, since, in practice, these preconditioners are not applicable to larger SANs. We also include the number of nonzeros needed to store the infinitesimal generator in some sparse matrix format ($nnz(Q)$) as well in the compact, storage-efficient SAN format ($nnz(SAN)$). Table 4 displays the comparisons for the largest size of each SAN example tested. For example, the results for san2 are for the SAN of order 1,024.

6.2.2. Number of Iterations Until Convergence. Now that NKP preconditioner's minimal storage requirements are evident, it remains to establish its effectiveness in terms of the number of iterations required for convergence. We compare the various SAN preconditioners in terms of the number of iterations and time until convergence for the three iterative methods: the power method, GMRES, and BiCGSTAB.

We present only two example SANs, san3B and san5B. Example san3B is representative of our example SANs, while san5B is the most complex SAN we tested. We summarize the results for the remaining five SANs in a single table (Table 7) toward the end of this section.

Example san3B. In this three-server queueing network, customers arrive to station 1 according to a Poisson distribution with parameter λ_1 and station 2 with parameter λ_2. With probability p_1, the customers from station 1 go to station 3. Thus, with probability $1 - p_1$, they exit the system. With probability p_2, the customers from station 2 go to station 3. With probability $1 - p_2$, they exit the system. Customers arriving at a full station are lost rather than blocked. The exponential service rates at each station are μ_1, μ_2, and μ_3, respectively. See Figure 3.

Three stochastic automata ($N = 3$) can be used to model this system, which has two synchronizing events ($E = 2$) and no functional transition rates. Thus, $T = 2E + N = 7$ and $NT = 21$. We choose the following parameters:

$$\lambda_1 = 15, \quad \lambda_2 = 10, \quad \mu_1 = 11, \quad \mu_2 = 12, \quad \mu_3 = 5,$$
$$p_1 = 0.7, \quad p_2 = 0.4,$$

and vary the capacities of the queues (C_1, C_2, and C_3) in order to create models of various sizes.

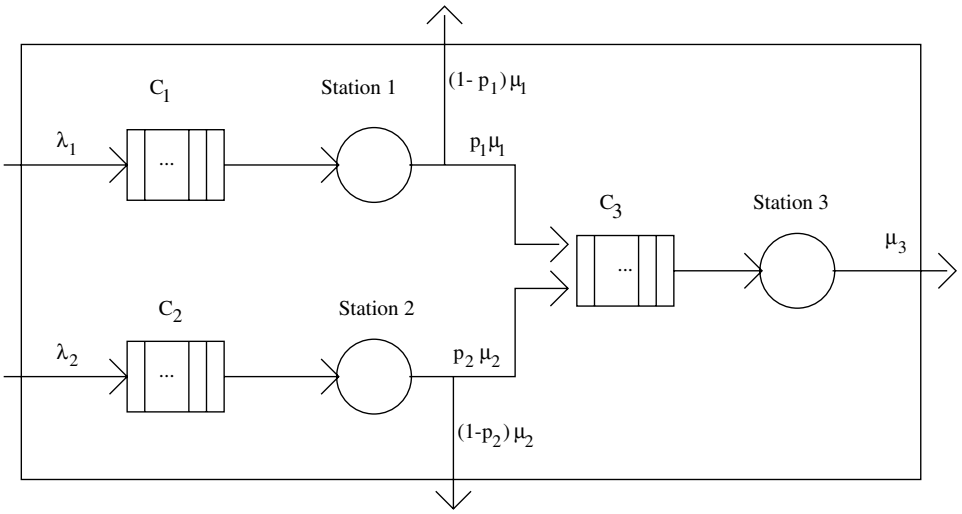

Figure 3 Example san3B**: Queueing Network with $N = 3$ Automata**

Table 5 Number of Iterations and CPU Times for Stationary Analysis of `san3B`

					Power		GMRES		BiCGSTAB	
C_1	C_2	C_3	Order	Preconditioner	Iter.	Time	Iter.	Time	Iter.	Time
3	3	4	80	None	262	23.01	9	9.97	28	6.43
				Neumann	88	23.08	4	12.18	12	7.88
				Indiv. Inv.	165	23.15	4	6.71	13	4.04
				Diagonal	184	16.17	7	7.74	23	4.31
				NKP	81	8.53	4	5.18	13	2.98
				*ILU*0	40	3.57	3	3.34	10	2.02
				ILUTH	41	3.66	3	3.34	9	1.84
6	6	8	441	None	492	180.46	14	70.52	55	46.95
				Neumann	165	181.48	7	92.09	25	58.43
				Indiv. Inv.	328	192.48	6	44.74	25	30.85
				Diagonal	392	143.58	13	65.88	45	34.55
				NKP	174	76.88	6	34.80	25	23.98
				*ILU*0	84	33.59	4	21.42	15	16.13
				ILUTH	115	46.39	5	26.98	17	17.36
9	9	9	1,000	None	877	678.36	20	227.25	74	117.93
				Neumann	293	683.30	11	309.32	34	162.66
				Indiv. Inv.	683	885.97	11	181.87	33	84.77
				Diagonal	746	591.55	17	194.55	63	100.70
				NKP	340	365.04	9	118.75	29	56.40
				*ILU*0	144	135.81	6	80.82	19	61.24
				ILUTH	165	156.45	6	79.18	23	60.12

We find the stationary solution of this SAN by using the three iterative methods (power, GMRES, and BiCGSTAB) with each preconditioner (no preconditioner, Neumann preconditioner, preconditioner based on individual inverses, diagonal preconditioner, NKP preconditioner, *ILU*0 preconditioner, and *ILUTH* preconditioner). Table 5 shows the results of using the various preconditioners on this three-dimensional example for models of various sizes. For the Neumann preconditioner, $H = 2$. For all the examples in this section, the programs were run on a Sun Ultra 10 with the convergence criterion that the maximum norm of the residual vector be less than 10^{-8}. The times reported in Table 5 do not include the time required to pre-compute each preconditioner, if necessary. A discussion of these preconditioner computation times is in §6.2.3.

Example `san5B`. We hypothesize that the SAN NKP preconditioner might struggle as N, the number of automata, or E, the number of synchronizing events, grows. The nonlinear minimization involves more variables as N and E grow, making the optimization problem more difficult. Thus, we increase the number of automata and synchronizing events in the five-dimensional examples to test the applicability of the NKP preconditioner truly.

The queueing network for example `san5B` contains five automata and four synchronizing events; see Figure 4. Five stochastic automata ($N = 5$) can be used to model this system, which has four synchronizing events ($E = 4$) and no functional transition rates. Thus, $T = 2E + N = 13$ and $NT = 65$. We choose the following parameters:

$$\lambda_1 = 10, \quad \lambda_2 = 11, \quad \mu_1 = 7, \quad \mu_2 = 8, \quad \mu_3 = 9,$$
$$\mu_4 = 9, \quad \mu_5 = 10,$$

and vary the capacities of the queues (C_1, C_2, C_3, C_4, and C_5) in order to create models of various sizes. Table 6 shows the results of using the various precon-

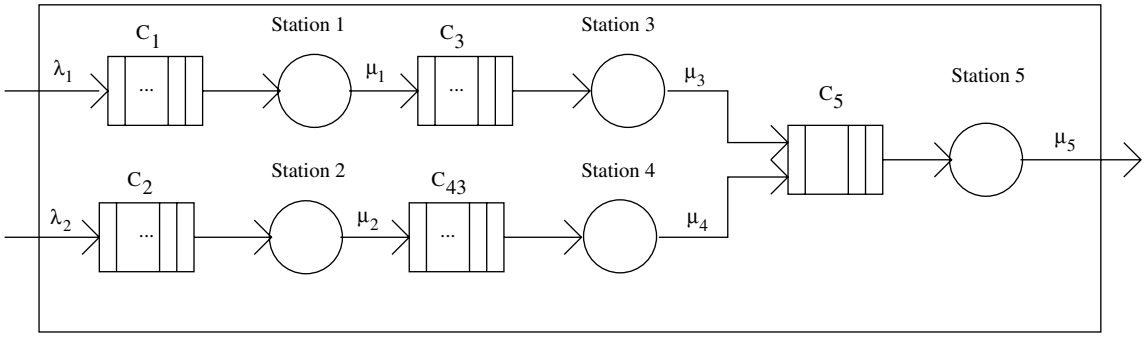

Figure 4 Example `san5B`: **Queueing Network with $N = 5$ Automata**

Table 6 Number of Iterations and CPU Times for Stationary Analysis of `san5B`

							Power		GMRES		BiCGSTAB	
C_1	C_2	C_3	C_4	C_5	Order	Preconditioner	Iter.	Time	Iter.	Time	Iter.	Time
1	1	2	2	2	108	None	144	66.38	8	41.99	30	27.11
						Neumann	48	67.15	4	61.56	13	36.86
						Indiv. Inv.	96	68.77	5	40.26	15	22.00
						Diagonal	261	123.34	7	37.01	27	25.03
						NKP	57	29.18	4	23.26	16	16.69
						*ILU*0	25	11.68	3	15.80	7	6.95
						ILUTH	19	8.82	3	15.79	7	6.93
2	2	2	3	3	432	None	273	437.67	11	203.08	43	138.83
						Neumann	91	418.16	5	270.05	19	182.48
						Indiv. Inv.	156	378.43	6	165.40	21	102.92
						Diagonal	438	675.85	10	178.18	41	127.35
						NKP	103	174.25	6	117.34	21	76.37
						*ILU*0	40	65.00	3	53.75	10	34.97
						ILUTH	37	57.60	3	56.68	9	30.91
3	3	3	3	3	1,024	None	301	988.21	12	487.73	57	407.29
						Neumann	100	983.18	6	676.42	21	454.29
						Indiv. Inv.	173	884.69	6	362.54	27	281.89
						Diagonal	375	1,257.41	11	440.88	50	339.30
						NKP	123	463.85	7	309.91	27	201.03
						*ILU*0	49	168.73	4	169.12	12	97.98
						ILUTH	47	161.56	4	167.67	12	101.39

ditioners on this five-dimensional example for models of various sizes.

Figures 5 and 6 summarize the number of iterations and time required by BiCGSTAB on the three models of `san5B` with sizes 108, 432, and 1,024. For all examples and all methods, the graphs show the same trend. The NKP preconditioner requires about as many iterations as does the Neumann or individual inverse preconditioners yet it overcomes the downfall of these two preconditioners as it requires the least CPU time compared with all other SAN preconditioners.

Summary Table for All Seven SANs: Table 7 summarizes the results for all seven SAN examples. For each SAN example, the results for the largest size model (order $\approx 1{,}000$) are reported.

6.2.3. Computation Time for Obtaining the NKP Preconditioner. By far, the most time-consuming step of the NKP preconditioner's computation is Step 2 of §6.1: solving the nonlinear minimization problem (1). We use SAS's `proc nlp` to solve for the NT unknowns. In all seven examples, SAS found a solution on a Sun Ultra 10 in less than six seconds, which is about the time required by ten power iterations. The less complex examples took less than one second. As the complexity of the SAN grows (that is, as N and T grow), the minimization problem becomes more difficult, requiring more time. However, we note that after grouping (Plateau et al. 2001), six seconds may be an upper bound on the time required to find the NT unknowns. Thus, the NKP preconditioner scales up nicely. The interesting part of minimization equation (1) is that for fixed N and T, the time required by the nonlinear optimization tool remains constant regardless of the size of the individual automata. Suppose that the N and T of example `san5B` are fixed at five and 13, respectively, and that larger models are run. For example, suppose the order of the global SAN grows from order 1,024 to order 10^{10}. In example `san5B`, this means $C_i = 99$ rather than $C_i = 3$ for $i = 1, \ldots, 5$. This minimization of (1) still has $NT = 65$ unknowns and still is solved in less than six seconds. The size of the individual automata does not change the minimization problem, and only affects the calculation of the traces, which remains negligible for the 100×100 matrices corresponding to the individual automata. Also, `san5B` is representative of larger SAN examples in terms of the number of automata used. Rarely, in practice is $N > 5$ since grouping has become a well-documented advantageous strategy (Plateau et al. 2001).

6.2.4. Summary of the Examples. Several observations are apparent from perusal of these tables and graphs.

1. The NKP preconditioner enjoys considerable success as a SAN preconditioner. Recall its dismal failure as a MC preconditioner. However, when the MC is represented as a SAN, the NKP is, in fact, the best possible choice for a preconditioner. (Note that the *ILU* preconditioners are used only for comparison purposes. In practice, *ILU* preconditioners are not applicable to SANs, due to their Kronecker structure. In these small examples, we first formed the global Q as a two-dimensional matrix, then

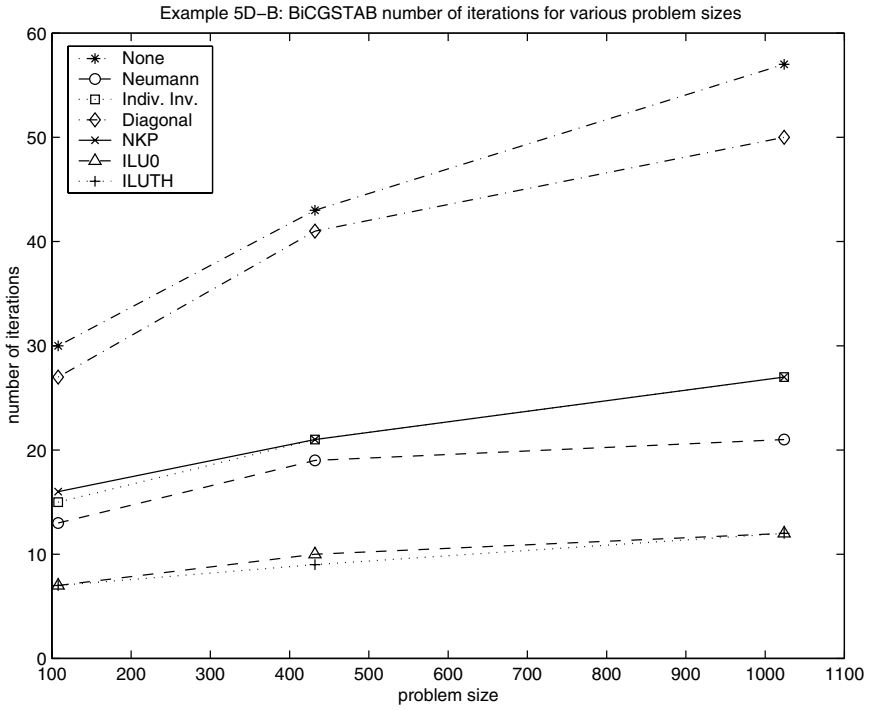

Figure 5 Number of Iterations Versus Model Size for `san5B` Using BiCGSTAB Method

formed the *ILU* preconditioner. Such as luxury is not available for larger SANs.) The NKP preconditioner clearly outperforms all the current SAN preconditioners in terms of usability (and at times performs nearly as well as the popular, though inapplicable, *ILU* preconditioners). Not only does the NKP preconditioner reduce the number of iterations required until convergence to the stationary solution, but it also reduces the time, which is the difficulty with the theoretically sound yet impractical Neumann and individual-inverse preconditioners. At each iteration, the NKP preconditioned methods require only a

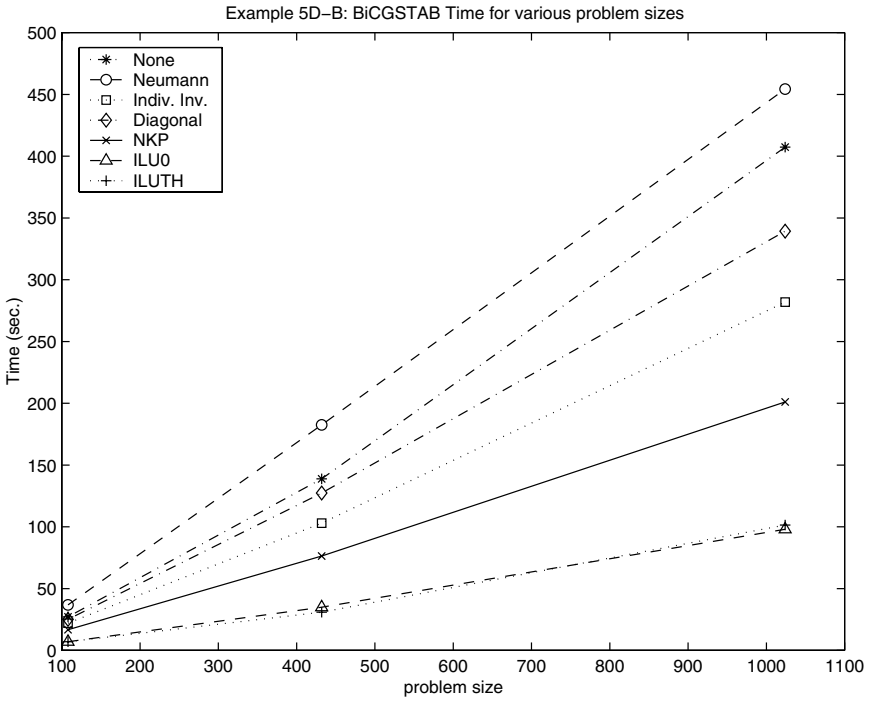

Figure 6 Time Versus Model Size for `san5B` Using BiCGSTAB Method

Table 7 Summary of Seven SANs: Number of Iterations and Time

			Power		GMRES		BiCGSTAB	
SAN	Order	Preconditioner	Iter.	Time	Iter.	Time	Iter.	Time
san2	1,024	None	6,992	317.34	61	80.88	222	35.95
		Neumann	2,329	315.16	32	93.15	85	34.52
		Indiv. Inv.	5,326	710.73	29	72.24	88	28.24
		Diagonal	6,650	301.68	62	84.70	214	29.47
		NKP	2,975	268.95	48	60.41	93	21.16
		*ILU*0	1,350	271.23	30	45.93	51	23.11
		ILUTH	557	131.94	19	37.26	36	18.98
san3A	1,000	None	861	655.04	21	240.17	89	140.57
		Neumann	288	655.54	8	227.64	28	130.84
		Indiv. Inv.	584	711.77	10	164.90	38	94.31
		Diagonal	771	587.85	18	207.20	77	121.27
		NKP	176	161.59	9	118.88	38	71.40
		*ILU*0	165	158.96	6	80.64	20	61.51
		ILUTH	126	120.34	5	66.23	19	60.50
san3B	1,000	None	877	678.36	20	227.25	74	117.93
		Neumann	293	683.30	11	309.32	34	162.66
		Indiv. Inv.	683	885.97	11	181.87	33	84.77
		Diagonal	746	591.55	17	194.55	63	100.70
		NKP	340	365.04	9	118.75	29	56.40
		*ILU*0	144	135.81	6	80.82	19	61.24
		ILUTH	165	156.45	6	79.18	23	60.12
san4A	1,000	None	648	957.77	18	297.16	67	168.74
		Neumann	216	920.51	7	304.08	24	181.06
		Indiv. Inv.	414	996.03	8	197.09	27	107.57
		Diagonal	489	735.01	14	233.79	51	126.06
		NKP	235	371.99	7	136.41	23	68.96
		*ILU*0	110	286.43	5	92.39	16	56.62
		ILUTH	104	271.60	5	90.15	16	55.03
san4B	1,000	None	388	662.16	14	303.64	60	208.66
		Neumann	130	667.76	6	356.92	20	210.87
		Indiv. Inv.	272	730.43	7	227.63	30	164.30
		Diagonal	302	515.77	12	262.45	54	186.55
		NKP	166	324.17	7	171.67	25	99.82
		*ILU*0	73	137.27	5	119.50	16	76.09
		ILUTH	62	116.96	4	94.62	18	85.95
san5A	1,024	None	335	617.17	12	282.10	51	188.72
		Neumann	112	615.31	5	320.11	21	236.29
		Indiv. Inv.	208	612.21	6	239.52	22	132.01
		Diagonal	208	383.57	9	215.01	42	155.78
		NKP	106	236.23	5	135.06	20	84.10
		*ILU*0	56	111.27	4	100.62	12	57.07
		ILUTH	56	109.35	4	99.02	14	66.07
san5B	1,024	None	301	988.21	12	487.73	57	407.29
		Neumann	100	983.18	6	676.42	21	454.29
		Indiv. Inv.	173	884.69	6	362.54	27	281.89
		Diagonal	375	1,257.41	11	440.88	50	339.30
		NKP	123	463.85	7	309.91	27	201.03
		*ILU*0	49	168.73	4	169.12	12	97.98
		ILUTH	47	161.56	4	167.67	12	101.39

vector-descriptor multiplication followed by a vector-Kronecker product multiplication.

2. The NKP preconditioner is storage-efficient due to the use of Kronecker products. The impractical Neumann preconditioner does not require any additional storage beyond the storage of Q, but it is too computation-intensive. The diagonal preconditioner similarly requires no additional storage and provides little additional computational burden, yet provides little, if any, preconditioning power. The individual inverse preconditioner requires slightly more storage than the NKP preconditioner and provides far less preconditioning power.

3. The computation of the NKP preconditioner is negligible. The maximum time required by SAS's `proc nlp` for solving the minimization problem of (1)

is less than six seconds, which is equivalent to the time required to do about ten iterations of the power method. The time required by SAS is dependent on the size of the minimization problem. Thus, as N and E grow, the minimization over $NT = N(2E + N)$ variables becomes more time-consuming. Adjusting `proc nlp`'s parameters and allowing the algorithm to run longer gives better solutions in the sense that the NKP is closer to Q, reducing $\|Q - NKP\|_F$ and thus making the NKP preconditioner more effective in reducing the number of iterations for well-conditioned SANs.

4. Our largest, most complex example, `san5B`, with $N = 5$ automata and $E = 4$ synchronizing events is a very representative example. In this case, $T = 2E + N = 13$. The minimization of (1) has $NT = 65$ variables and is solved in less than six seconds on a Sun Ultra 10. Suppose $N = 5$ and $T = 13$ remain fixed, but larger models are run. For example, suppose that the order of the global MC grows from order 1,024 to order 10^{10}. For `san5B`, this means $C_i = 99$ rather than $C_i = 3$ for $i = 1, \ldots, 5$. This minimization of (1) still has $NT = 65$ unknowns and still is solved in less than six seconds despite the fact that the global order has grown from about 10^3 to 10^{10}. The size of the individual automata does not change the minimization problem, but it only affects the calculation of the traces, which remains negligible for the 100×100 matrices corresponding to the individual automata. Also, `san5B` is representative of larger SAN examples in terms of the number of automata used. Rarely, in practice is $N > 5$ since grouping has become a well-documented advantageous strategy (Plateau et al. 2001). One final reason why we claim that `san5B` is very representative concerns the conditioning of linear systems. We expect the result that the NKP preconditioner beats all existing SAN preconditioners in terms of both number of iterations and overall time to remain as larger and larger examples are tested. Ipsen and Meyer (1994) and O'Cinneide (1993) have shown that MCs are well-conditioned, meaning that small perturbations in Q cannot have a drastic effect on the stationary solution. As the size of the models increase, we do not expect to encounter conditioning problems. In fact, for `san5B`, the two-norm of the difference between the exact solution and the computed NKP preconditioned solution is 2.5173×10^{-12}, which suggests that the problem is well-conditioned.

6.3. Conclusions of the NKP Preconditioner Applied to SANs

After the NKP preconditioner's dismal performance on MCs, its improvement when applied to SANs is marked. This drastic improvement underscores the need to account for structure in a particular problem. When the MC is represented as a SAN rather than a matrix, the NKP preconditioner becomes the best current preconditioner, reducing both the number of iterations and time until convergence to the stationary solution while requiring minimal additional storage and computation time.

7. Future Work

Much more extensive testing needs to be done to verify the NKP preconditioner's success and to establish the range of SAN problems for which the NKP is most applicable. We provide the following suggestions for future work.

1. Test nonqueueing examples. All examples in this paper are derived from queueing networks. A variety of examples from different disciplines would make a more well-rounded study.

2. Test on larger SANs such as size 100 automata with $N = 5$. The size of the problems we tested was limited by MATLAB's memory requirements and slow speed, problems easily eliminated by the use of programming languages like Fortran or C. However, we emphasize that we fully expect our results to hold as the problem size increases, since the properties and structure of the SAN, which have made the NKP preconditioner so successful, obviously remain unchanged.

3. Compute the higher-order singular value decomposition (HOSVD) of Q so that the approximations, $A_1 \approx \alpha_1 Q_1^{(1)} + \alpha_2 Q_2^{(1)} + \cdots + \alpha_T Q_T^{(1)}$, $A_2 \approx \beta_1 Q_1^{(2)} + \beta_2 Q_2^{(2)} + \cdots + \beta_T Q_T^{(2)}$, and $A_N \approx \eta_1 Q_1^{(N)} + \eta_2 Q_2^{(N)} + \cdots + \eta_T Q_T^{(N)}$, can be replaced with equations. Determine if the computational expense required to compute the HOSVD of Q, which is expected to be quite high, is offset by the improved quality of the NKP.

4. Explore the use of parallel machines for SAN problems. Due to the SAN descriptor's structure $\sum_{j=1}^{T} \bigotimes_{i=1}^{N} Q_j^{(i)}$, parallel operations can be used in the vector-Kronecker product multiplication algorithm (Fernandes et al. 1998). Each of the $j = 1, \ldots, T$ terms in the vector-SAN descriptor multiplication of $x[\sum_{j=1}^{T} \bigotimes_{i=1}^{N} Q_j^{(i)}]$ can be computed in parallel. The k^{th} machine computes $x \bigotimes_{i=1}^{N} Q_k^{(i)}$. We predict significant time savings with parallelism since the vector-descriptor multiplications are the most computationally intensive step in most iterative methods.

5. Study the minimization problem of (1). Perhaps structure can be exploited to improve running times or the accuracy of solutions.

6. Incorporate the NKP preconditioner into SAN packages like PEPS (Plateau 1990).

8. Conclusion

We have thoroughly tested the new SAN preconditioner, the nearest Kronecker product (NKP) preconditioner, introduced in Langville and Stewart (2002).

We have verified the NKP preconditioner's initial success on small artificial SAN examples (Langville and Stewart 2002) by testing the NKP preconditioner extensively on larger SANs. We also explain the seemingly paradoxical finding of the NKP preconditioner's failure on MCs. We conclude that, while the NKP preconditioner for MCs is poor, the NKP preconditioner for SANs is the current best SAN preconditioner.

Acknowledgments

This research was supported in part by NSF (CCR-9731856).

References

Benzi, M., M. Tuma. 2002. A parallel solver for large-scale Markov chains. *Appl. Numer. Math.* **41** 135–153.

Brown, P. H. Walker. 1997. GMRES on (nearly) singular systems. *SIAM J. Matrix Anal.* **18** 37–51.

Buchholz, P. 1999. Projection methods for the analysis of stochastic automata networks. B. Plateau, W. J. Stewart, M. Silva, eds. *Numerical Solution of Markov Chains*. Prensas Universitarias de Zaragoza, Zaragoza, Spain, 149–168.

Buchholz, P., G. Ciardo, S. Donatelli, P. Kemper. 2000. Complexity of memory-efficient Kronecker operations with applications to the solution of Markov models. *INFORMS J. Comput.* **12** 203–222.

Cao, Z. 1997. A note on the convergence behavior of GMRES. *Appl. Numer. Math.* **25** 13–20.

Dayar, T. 1998. State space orderings for Gauss-Seidel in Markov chains revisited. *SIAM J. Sci. Comput.* **19** 148–154.

Fernandes, P., B. Plateau, W. J. Stewart. 1998. Efficient descriptor-vector multiplications in stochastic automata networks. *J. Association Comput. Machinery* **45** 381–414.

Freund, R. W., M. Hochbruck. 1994. On the use of two QMR algorithms for solving singular systems and applications in Markov chain modeling. *Numer. Linear Algebra Appl.* **1** 1–7.

Golub, G. H., F. Luk, M. Overton. 1981. A block Lanczos method for computing the singular values and corresponding singular vectors of a matrix. *ACM Trans. Math. Software* **7** 149–169.

Greenbaum, A. 1997. *Iterative Methods for Solving Linear Systems*. SIAM, Philadelphia, PA.

Ipsen, I. C. F. 2000. Expressions and bounds for the residual in GMRES. *BIT* **40** 524–533.

Ipsen, I. C. F., C. D. Meyer. 1994. Uniform stability of Markov chains. *SIAM J. Matrix Anal. Its Appl.* **15** 1061–1074.

Krieg, G. N., H. Kuhn. 2002. A decomposition method for multi-product kanban systems with setup times and lost sales. *IEE Trans.* **34** 613–625.

Langville, A. N. 2002. Preconditioning for Stochastic Automata Networks. Ph.D. thesis, Operations Research Program, North Carolina State University, Raleigh, NC.

Langville, A. N., W. J. Stewart. 2002. A Kronecker product approximate inverse preconditioner for SANs. *Numer. Linear Algebra Appl.* Forthcoming.

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

Philippe, B., Y. Saad, W. J. Stewart. 1992. Numerical methods in Markov chain modeling. *Oper. Res.* **40** 1156–1179.

Pitsianis, N., C. Van Loan. 1993. Approximation with Kronecker products. M. S. Moonen, G. H. Golub, eds. *Linear Algebra for Large Scale and Real Time Applications*. Kluwer Publications, Dordrecht, The Netherlands, 293–314.

Plateau, B. 1985. On the stochastic structure of parallelism and synchronization models for distributed algorithms. *Performance Evaluation Rev.* **13** 142–154.

Plateau, B. 1990. PEPS: Package for solving complex Markov models of parallel systems. R. Puigjaner, D. Potier, eds. *Modelling Techniques and Tools for Computer Performance Evaluation*. Plenum Press, New York, 291–306.

Plateau, B., J. M. Fourneau. 1991. A methodology for solving Markov models of parallel systems. *J. Parallel Distributed Comput.* **12** 370–387.

Plateau, B., W. J. Stewart, P. Fernandes. 2001. On the benefits of using functional transitions in Kronecker modelling. *Performance Evaluation*. Forthcoming.

Rouskas, G. N., A. Halim Zaim, H. G. Perros. 2002. Computing call blocking probabilities in LEO satellite networks: The single orbit case. *IEEE Trans. Vehicular Technology* **51** 332–347.

Saad, Y. 1995. Preconditioned Krylov subpace methods for the numerical solution of Markov chains. W. J. Stewart, ed. *Computations with Markov Chains*. Kluwer Academic. Boston, MA, 49–64.

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

Stewart, W. J. 2003. http://www.csc.ncsu.edu/faculty/WStewart/index.html.

Stewart, W. J., K. Atif, B. Plateau. 1995. The numerical solution of stochastic automata networks. *Eur. J. Oper. Res.* **86** 503–525.

Toh, K. 1997. GMRES vs. ideal GMRES. *SIAM J. Matrix Anal. Appl.* **18** 30–36.

Uysal, E., T. Dayar. 1998. Iterative methods based on splittings for stochastic automata networks. *Eur. J. Oper. Res.* **110** 166–186.