A Kronecker product approximate preconditioner for SANs

Amy N. Langville[1,*,†] and William J. Stewart[2,‡]

[1]*Department of Mathematics, N. Carolina State University, Raleigh, NC 27695-8205, U.S.A.*
[2]*Department of Computer Science, N. Carolina State University, Raleigh, N.C. 27695-8206, U.S.A.*

SUMMARY

Many very large Markov chains can be modelled efficiently as stochastic automata networks (SANs). A SAN is composed of individual automata which, for the most part, act independently, requiring only infrequent interaction. SANs represent the generator matrix Q of the underlying Markov chain compactly as the sum of Kronecker products of smaller matrices. Thus, storage savings are immediate. The benefit of a SAN's compact representation, known as the descriptor, is often outweighed by its tendency to make analysis of the underlying Markov chain tough. While iterative or projections methods have been used to solve the system $\pi Q = 0$, the time until these methods converge to the stationary solution π is still unsatisfactory. SAN's compact representation has made the next logical research step of preconditioning thorny. Several preconditioners for SANs have been proposed and tested, yet each has enjoyed little or no success. Encouraged by the recent success of approximate inverses as preconditioners, we have explored their potential as SAN preconditioners. One particularly relevant finding on approximate inverse preconditioning is the nearest Kronecker product approximation discovered by Pitsianis and Van Loan. In this paper, we extend the nearest Kronecker product technique to approximate the Q matrix for an SAN with a Kronecker product, $A_1 \otimes A_2 \otimes \cdots \otimes A_N$. Then, we take $M = A_1^{-1} \otimes A_2^{-1} \otimes \cdots \otimes A_N^{-1}$ as our SAN NKP preconditioner. Copyright © 2004 John Wiley & Sons, Ltd.

KEY WORDS: stochastic automata networks; nearest Kronecker products; multilinear algebra; preconditioning

1. INTRODUCTION

A very large Markov chain has a very large state space requiring extensive storage and making analysis difficult. Many large Markov chains can be effectively defined and analysed using the newer model of stochastic automata networks (SANs). SANs represent the generator matrix Q compactly; thus storage savings are immediate. As long as the analysis of the SAN

---

*Correspondence to: Amy Langville, Mathematics Department, N. Carolina State University, Raleigh, NC 27695-8205, U.S.A.
†E-mail: anlangvi@unity.ncsu.edu
‡E-mail: billy@csc.ncsu.edu

remains reasonable, we have realized significant savings over the Markov chain model with its state space explosion problem.

SANs were first proposed by Plateau [1] in 1985 and have been actively researched since. A SAN is a collection of individual stochastic automata that generally act independently of one another, requiring only infrequent interaction. Given this description, it is easy to see why SANs have been successfully applied to parallel systems such as shared memory or communicating processes. Each automaton is represented by a set of states and transitions which model the movement from one state to another. The state of an automaton at time t is given by the state it occupies at time t. The global state, the state of the SAN, at time t is the state of each of its automata.

The global state may change when a transition occurs. Transitions can be either local or synchronizing. Local transitions only affect the corresponding automata. When an automata has a local transition, it moves from one of its states to another of its states. Synchronizing transitions are not local. They affect the global state by changing the state of several automata. A synchronizing transition occurs when one automaton enables a transition to occur in two or more other automata.

We also make another distinction between transitions. They can be either constant or functional. A functional transition occurs when an automaton's transition rate is a function of the state of another automaton. Transitions that are not functional are called constant. Constant or functional transitions, unlike synchronizing transitions, affect only the local automata involved. Note that synchronizing transitions may be constant or functional. This information regarding the automata and their types of transitions provides all the information needed to formally define a SAN, as Atif and Plateau have done [2]. While this infrequent interaction (synchronizing transitions and functional transition rates) does complicate SANs, Plateau and her co-workers have shown that the SAN can still be represented in compact form as a sum of Kronecker products, known as the SAN descriptor [1, 3, 4]. Of course, too much interaction may complicate the SAN model to the point that all savings have been lost. Therefore, SAN models should be restricted to systems with appropriate infrequent interaction.

As emphasized above, SANs with their compact representation (the descriptor) clearly save storage in the modelling of parallel systems, which have automata acting independently for the most part with only occasional interaction. But storage savings alone are useless if model analysis cannot be done efficiently. One type of model analysis involves determining the transient probability distribution for the system. Another type of analysis determines the system's stationary probability distribution. It is this type of analysis that we study in the present project. Since SANs are intimately tied to Markov chains, we begin by studying the stationary solution techniques for Markov chains. There are three primary classes of solution techniques: direct methods, iterative methods and projection methods. 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, U factors. Furthermore, SANs are used as a compact, alternative representation for very large Markov models. The size of such models makes direct methods impractical [4]. To circumvent the state space explosion problem associated with these large systems, Atif and Buchholz have defined equivalence relations for SANs, thereby providing an equivalent SAN of smaller size [5, 6]. Nevertheless, direct methods are not the methods of choice for SANs.

In contrast, iterative and projection methods for solving linear systems have been studied extensively, and some of these methods have been successfully applied to SANs. In solving

a SAN for the stationary solution, one can always expand the information for local automata and transitions to create the two-dimensional global generator matrix and then apply a standard iterative method to the global Markov model. Philippe, Saad, Stewart and Wu discuss numerical methods for Markov chains in [4, 7, 8]. However, since the value of SANs lies in their storage-saving, compact representation of the Markov model, researchers rarely use expansion to obtain the two-dimensional global generator matrix. This would be counter to the purpose of using a SAN. For example, ten stochastic automata, each with ten states and a dense transition matrix, result in an expanded global matrix with 10^{10} elements [4]. This example explains why research has focused on computation of the global stationary probabilities by using the individual automata information contained in the descriptor without ever generating the two-dimensional global matrix.

Toward this end, the classical iterative methods of Gauss–Seidel, and SOR as well as the aggregation/disaggregation methods cannot be easily used [8], although Dayar has done some work with the Gauss–Seidel method [9, 10]. However, 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 Reference [11]. These methods include the power method and projection methods such as GMRES and Arnoldi. The GMRES and Arnoldi methods have been found to outperform the power method when applied to SANs [12]. The previously considered projection methods, GMRES and Arnoldi, both use long recurrences. This requires a large number of vectors to be stored, thereby increasing the work at each iteration. Projection methods with shorter recurrences, namely BiCGSTAB, CGS and TFQMR have also been used [13]. Without preconditioners, BiCGSTAB, CGS and TFQMR seemed comparable to GMRES and Arnoldi in terms of iteration count and running time. Preconditioned BiCGSTAB, CGS and TFQMR generally outperformed preconditioned GMRES and Arnoldi.

As SANs are so closely related to Markov chains, studying techniques that accelerate convergence of iterative methods applied to Markov chains might also help to accelerate convergence in SANs. Preconditioning is a popular means of achieving improved convergence of iterative methods. Preconditioning for Markov models is reviewed in References [4, 7, 14]. Preconditioners based on incomplete LU factorizations, which have been successfully employed in Markov models, cannot be readily used for SANs for the same reasons that direct methods based on L, U factors cannot be used [12]. A SAN preconditioner based on the Neumann series was proposed in Reference [12]. Yet such a preconditioner is computationally too expensive to compute, as experiments showed. While the number of iterations needed for convergence decreased, the running time per iteration increased. In fact, the results presented in Reference [12] showed that no real benefit was derived from that particular preconditioner, emphasizing the need for better preconditioners. Preconditioners have also been applied to the projection methods of BiCGSTAB, CGS, TFQMR [13]. These preconditioners, which were extensions of the Neumann preconditioner, required less work to generate and resulted in a reduction in overall solution time. However, much more work remains to be done to develop efficient preconditioners, especially for very large systems.

Our aim is to address this need by examining the plausibility of one particular approximate preconditioner, the nearest Kronecker product 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 Markov chain represented as a SAN. The results can also be applied to certain methods for computing transient solutions. Throughout this paper, π is a row vector representing the stationary solution of a continuous-time ergodic Markov chain. Unless stated otherwise, all other vectors are column vectors. The column vector e is the vector of all 1's.

In general, the computation of the stationary solution π of a continuous-time ergodic Markov chain involves solving the linear system $\pi Q = 0$ and $\pi e = 1$, where Q is the infinitesimal generator of the Markov chain and e represents the unit row vector. Q is singular with rank $n - 1$. Thus, finding the stationary solution of a continuous-time Markov chain can be viewed as a linear system problem. Another way to view the same problem is as an eigenvalue problem. P is the transition probability matrix associated with the same system. In fact, $P = I + \Delta t Q$ where $\Delta t \leqslant 1/\max |q_{ii}|$. P is a stochastic matrix with a unit eigenvalue. Then finding the stationary solution π involves solving $\pi = \pi P$, which is an eigenvalue problem. Now this eigenvalue problem can be used to define the power method, an iterative method for finding π by computing iterates with

$$x^{(k+1)} = x^{(k)} P$$

With a suitable initial row vector iterate $x^{(0)}$, $x^{(k+1)}$ will converge to the eigenvector π which can then be normalized so that π contains the stationary solution.

Very large Markov chains are often represented as SANs using the SAN descriptor in place of Q. Namely, $Q = \sum_{j=1}^{T} \otimes_{i=1}^{N} Q_j^{(i)}$, where $T = 2E + N$, E is the number of synchronizing events and N is the number of automata. Since $P = I + \Delta t Q$, then in the SAN formalism,

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

and the power method for SANs can be written as

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

The power method is the simplest of all iterative methods for finding the stationary solution vector π. The Jacobi, Gauss–Seidel and SOR method are three more iterative methods used for solving linear systems, such as our homogeneous linear system $\pi Q = 0$. Yet these methods are based on splittings of the transition matrix and thus are not easily transferable to the SAN formalism. 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 QMR. Such projection methods can be and have been applied to SANs [12, 13, 15].

3. PRECONDITIONING

It is well known that the iterative methods discussed above perform better when precondi-tioners are used. The convergence of an iterative method depends on the eigenvalues of the system. Any iterative method can converge slowly if the eigenvalue distribution is undesir-able for that method. For example, when the subdominant eigenvalue of the iteration matrix is close to the dominant eigenvalue (which is 1 for our transition matrices P), the power method converges slowly. Thus, 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 $Ay = b$, we introduce the preconditioning matrix M, so that $MAy = Mb$. We hope that M is a good approximation of A^{-1} and thus convergence will be rapid.

For Markov chain problems, the preconditioned power method becomes

$$x^{(k+1)} = x^{(k)}(I - (I - P)M)$$

Since the matrix $(I - P)$ is singular with rank $(n-1)$, M is chosen to be a good approximation of a generalized inverse of $(I - P)$ [16]. Thus for SANs, the preconditioned power method is

$$x^{(k+1)} = x^{(k)}(I - (I - (I + \Delta t Q))M)$$

$$= x^{(k)}(I + \Delta t Q M)$$

$$= x^{(k)} + \Delta t x^{(k)} Q M$$

$$= x^{(k)} + \Delta t x^{(k)} \left(\sum_{j=1}^{T} \otimes_{i=1}^{N} Q_j^{(i)} \right) M$$

The problem now becomes that of finding a suitable preconditioner M that fits nicely into the SAN formalism. A popular set of preconditioners, *ILU* preconditioners, have largely been dismissed from consideration. Recall that the problem with adapting *ILU* preconditioners to SANs (for use in an iterative method, like the preconditioned power method) is that they are based on incomplete *LU* factorizations of the transition matrix. SANs store the transition matrix information as a sum of Kronecker products. And thus, an *LU* factorization of a SAN descriptor is not easily accessible.

3.1. A SAN preconditioner based on the Neumann series

In Reference [12], Stewart *et al.* introduce a preconditioner based on the Neumann series. Since $P = I + \Delta t Q$, then $(I - P) = -\Delta t Q$. It has been shown in Reference [17] that

$$(I - P)^{\#} = \sum_{h=0}^{\infty} (P^h - L)$$

where L is an $n \times n$ matrix with all rows equal to the stationary solution vector π and $(I - P)^{\#}$ is the group inverse of $(I - P)$. An approximation of $(I - P)^{\#}$ and thus M can be computed

by summing over the first H terms and substituting the most recent approximation of π, given in $x^{(k)}$, into each row of L. The preconditioned power method for SANs becomes

$$x^{(k+1)} = x^{(k)} + \Delta t x^{(k)} Q \left(\sum_{h=0}^{H} P^h - \tilde{L} \right)$$

$$= x^{(k)} + \Delta t x^{(k)} Q \left(\sum_{h=0}^{H} P^h \right) - \Delta t x^{(k)} Q \tilde{L}$$

where $\tilde{L}_{jl} = \pi_l^{(k)}$ for all j. Thus, $\tilde{L} = e\pi^{(k)}$ and hence $Q\tilde{L} = (Qe)\pi^{(k)} = 0$, since Q is an infinitesimal generator with row sums of zero. Thus, the preconditioned power method for SANs may be written as

$$x^{(k+1)} = x^{(k)} + \Delta t x^{(k)} Q \left(\sum_{h=0}^{H} P^h \right)$$

Writing Q and P as sums of Kronecker products gives

$$x^{(k+1)} = x^{(k)} + \Delta t x^{(k)} \left(\sum_{j=1}^{T} \otimes_{i=1}^{N} Q_j^{(i)} \right) \left(\sum_{h=0}^{H} \left(\otimes_{i=1}^{N} I_{n_i} + \sum_{j=1}^{T} \Delta t \otimes_{i=1}^{N} Q_j^{(i)} \right)^h \right)$$

By increasing H, the number of terms in the Neumann expansion, we can obtain a more accurate preconditioner and converge to the stationary solution in a smaller number of iterations. Yet at the same time, as H increases, clearly the computational effort needed to obtain this accurate preconditioner increases. Tests of this preconditioner on a queueing network model report that no real benefit was derived from this preconditioner [12].

3.2. A SAN preconditioner based on individual inverses

Another possibility explored by Buchholz [13] is the preconditioner which involves inverses of the individual automata matrices. This preconditioner is formed so that the repeated vector-Kronecker product multiplications which make the previous Neumann preconditioner so time consuming are avoided. Starting with the infinite sum, $\sum_{h=0}^{\infty} P^h$, Q is then split into three parts.

$$Q = Q_{\mathrm{D}} + Q_{\mathrm{L}} + Q_{\mathrm{S}}$$

Q_{D} is a diagonal matrix including all diagonal elements of Q. Q_{L} represents the local transitions and thus $Q_{\mathrm{L}} = \oplus_{i=1}^{N} Q_l^{(i)}$, where N is the number of automata and $Q_l^{(i)}$ represents the local transition matrix for automata i. Q_{S} represents the synchronizing transitions and thus $Q_{\mathrm{S}} = \sum_{j=1}^{T} \otimes_{i=1}^{N} Q_s^{(i)}$, where T is the number of synchronizing events and $Q_s^{(i)}$ describes the effect of synchronizing event j on automata i. Thus, since $P = I + \Delta t Q$,

$$\sum_{h=0}^{\infty} P^h = \sum_{h=0}^{\infty} (I + \Delta t Q_{\mathrm{D}} + \Delta t Q_{\mathrm{L}} + \Delta t Q_{\mathrm{S}})^h$$

After a great deal of algebraic and Kronecker manipulation [13], it can be shown that

$$\sum_{h=0}^{\infty} P^h \geqslant \left[\sum_{h=0}^{\infty} (I + \Delta t Q_D)^h \right] [\otimes_{i=1}^{N} (I - \Delta t Q_l^{(i)})^{-1}][I + \Delta t Q_S]$$

and hence a preconditioner can be defined as

$$M = N_D N_L (I + \Delta t Q_S)$$

where $N_D = \sum_{h=0}^{\infty} (I + \Delta t Q_D)^h$, which is a diagonal matrix with element (i,i) equal to $-1/\Delta t$ $Q_D(i,i)$ and $N_L = \otimes_{i=1}^{N} (I - \Delta t Q_l^{(i)})^{-1}$. This preconditioner consists of a diagonal matrix, the part of Q which corresponds to synchronizing transitions and a part which is the Kronecker product of inverses of individual automata matrices. These small $n_i \times n_i$ inverse matrices must be computed and stored as well as the diagonal of the preconditioner. The N_L matrix is a Kronecker product of usually dense matrices, requiring considerable computation time for the vector-Kronecker product multiplication needed at each iteration. While experiments showed that this preconditioner required fewer iterations until convergence than unconditioned methods, only in some cases did the overall solution times reduce.

3.3. Other SAN preconditioners

Other preconditioners for SANs have also been proposed recently by Plateau *et al.* [18]. One preconditioner is based on the additive Schwartz method. Incomplete *LU* factorizations of the individual Kronecker terms in the SAN descriptor are formed. The sum of the inverses of each of these individual Kronecker term matrices is then used as the preconditioning matrix. Another preconditioner known as multiplicative preconditioning is based on the multiplicative Schwartz method. In this case, the product, rather than the sum, of the inverses of each individual Kronecker term matrix is taken as the preconditioner. Numerical experiments with the additive preconditioner and the multiplicative preconditioner were unsuccessful. One more preconditioner, which has been studied, is the diagonal preconditioner. This, the simplest of all preconditioners, uses the inverse of the diagonal elements of Q to form a diagonal preconditioning matrix. This preconditioner provided interesting results. Numerical experimentation showed that simple diagonal preconditioning reduced the number of iterations without increasing the computation time per iteration!

3.4. An approximate inverse preconditioner: the nearest Kronecker product

Approximate inverse preconditioners have been successfully employed with iterative methods [12, 19–21]. Our approximate inverse preconditioner, the (Nearest Kronecker Product) NKP preconditioner, which is based on the Kronecker product approximations in Reference [22], will be discussed thoroughly in the next section. Pitsianis and Van Loan present a method for finding the nearest Kronecker product, $A \otimes B$, for a general matrix R [22]. Since $A \otimes B \approx R$, one would hope that $A^{-1} \otimes B^{-1} \approx R^{-1}$. They defined $A^{-1} \otimes B^{-1} = M$ as the preconditioner. On a small example, this preconditioner compared favourably with other preconditioners. This motivated us to extend this preconditioner to SANs. For Markov chains, we approximate $Q^{\#}$ rather than Q^{-1}. However, the algorithm in Reference [22] almost always gives a nonsingular A and nonsingular B. Thus, we use the standard inverses, A^{-1} and B^{-1}, to form the preconditioner. In effect, we are using the ideal preconditioner $M = A^{-1} \otimes B^{-1}$ for a nearby system

whose coefficient matrix $\hat{Q}$ is almost Q. Finding the small A^{-1} and B^{-1} matrices is not too difficult and the approximation is good enough for many matrices with inherent Kronecker structure to provide a preconditioner which reduces the number of iterations without adding much computational time. The Kronecker approximation for SANs avoids the formation of M, instead only A^{-1} and B^{-1} are required in the vector-Kronecker product multiplication of the iterative methods. We were able to extend Pitsianis and Van Loan's work to find any number of smaller matrices whose Kronecker product approximates the original matrix Q. Thus, we can find $A_1, A_2, \ldots, A_N$ such that $A_1 \otimes A_2 \otimes \cdots \otimes A_N \approx Q$. We take $M = A_1^{-1} \otimes A_2^{-1} \otimes \cdots \otimes A_N^{-1}$ as the NKP preconditioner for SANs. Our initial results with the NKP preconditioner for SANs look very promising. A closer look at the work of Pitsianis and Van Loan gives us insight into our quest for a good SAN preconditioner.

4. THE THEORY BEHIND THE NEAREST KRONECKER PRODUCT

Pitsianis and Van Loan find the nearest Kronecker product, $A \otimes B$, for an $m_1 m_2 \times n_1 n_2$ matrix R. They formulate their problem as follows: find matrices A, B so that $\|R - A \otimes B\|_F^2$ is minimized, where A is an $m_1 \times n_1$ matrix and B is an $m_2 \times n_2$ matrix. To this end, they define the matrix $\tilde{R}$, which is a special rearrangement of R, relative to the blocking parameters m_1, m_2, n_1 and n_2. We use a *vec*torizing operation which turns a matrix into a vector by stacking the columns of the matrix. For example, if

$$R = \begin{pmatrix} r_{11} & r_{12} & r_{13} & r_{14} \\ r_{21} & r_{22} & r_{23} & r_{24} \\ r_{31} & r_{32} & r_{33} & r_{34} \\ r_{41} & r_{42} & r_{43} & r_{44} \\ r_{51} & r_{52} & r_{53} & r_{54} \\ r_{61} & r_{62} & r_{63} & r_{64} \end{pmatrix}, \quad A = \begin{pmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{pmatrix}, \text{ and } B = \begin{pmatrix} b_{11} & b_{12} \\ b_{21} & b_{22} \\ b_{31} & b_{32} \end{pmatrix}$$

Then,

$$\tilde{R} = \begin{pmatrix} r_{11} & r_{21} & r_{31} & r_{12} & r_{22} & r_{32} \\ r_{41} & r_{51} & r_{61} & r_{42} & r_{52} & r_{62} \\ r_{13} & r_{23} & r_{33} & r_{14} & r_{24} & r_{34} \\ r_{43} & r_{53} & r_{63} & r_{44} & r_{54} & r_{64} \end{pmatrix}, \quad \text{vec}(A) = \begin{pmatrix} a_{11} \\ a_{21} \\ a_{12} \\ a_{22} \end{pmatrix}, \quad \text{vec}(B) = \begin{pmatrix} b_{11} \\ b_{21} \\ b_{31} \\ b_{12} \\ b_{22} \\ b_{32} \end{pmatrix}$$

If A is $m_1 \times n_1$, B is $m_2 \times n_2$, and R is $m_1 m_2 \times n_1 n_2$, then $\tilde{R}$ is $m_1 n_1 \times m_2 n_2$. To construct $\tilde{R}$, express the matrix R of dimension $m_1 m_2 \times n_1 n_2$ as $m_1 \times n_1$ blocks each of size $m_2 \times n_2$. That is,

$$R = \begin{pmatrix} R_{1,1} & R_{1,2} & \cdots & R_{1,n_1} \\ \vdots & \ddots & & \vdots \\ R_{m_1,1} & R_{m_1,2} & \cdots & R_{m_1,n_1} \end{pmatrix}$$

where $R_{i,j}$ is a block of size $m_2 \times n_2$. This partitioning of R depends on the choice of m_1 and n_1. Pitsianis and Van Loan form $\tilde{R}$ as

$$\tilde{R} = \begin{pmatrix} \text{vec}(R_{1,1})^{\mathrm{T}} \\ \text{vec}(R_{2,1})^{\mathrm{T}} \\ \vdots \\ \text{vec}(R_{m_1,1})^{\mathrm{T}} \\ \hline \text{vec}(R_{1,2})^{\mathrm{T}} \\ \text{vec}(R_{2,2})^{\mathrm{T}} \\ \vdots \\ \text{vec}(R_{m_1,2})^{\mathrm{T}} \\ \hline \vdots \\ \hline \text{vec}(R_{1,n_1})^{\mathrm{T}} \\ \text{vec}(R_{2,n_1})^{\mathrm{T}} \\ \vdots \\ \text{vec}(R_{m_1,n_1})^{\mathrm{T}} \end{pmatrix}$$

Using this rearrangement operation and the vectorizing operation, Pitsianis and Van Loan prove that

$$\|R - A \otimes B\|_F^2 = \|\tilde{R} - ab^{\mathrm{T}}\|_F^2$$

where $a = \text{vec}(A)$ and $b = \text{vec}(B)$ [22]. The original minimization problem is transformed into the problem of approximating a given matrix $\tilde{R}$ by a rank-1 matrix ab^{T}. The solution to this new problem is derived from the matrix singular value decomposition (SVD). The nearest rank-p matrix to a given matrix is the first p terms in the truncated SVD in dyadic form, $\sum_{i=1}^{p} \sigma_i U_i V_i^{\mathrm{T}}$, where σ_i is the ith largest singular value of $\tilde{R}$ and U_i, V_i are the corresponding left and right singular vectors. Thus, from the SVD of $\tilde{R}$, one realizes that $a = \sigma_1 U_1$ and $b = V_1$, where σ_1 is the largest singular value of $\tilde{R}$ and U_1, V_1 are the corresponding left and right singular vectors. The matrix $\tilde{R}$ never needs to be formed explicitly, only a vector-$\tilde{R}$ multiplication algorithm is needed. In fact, the necessary vectors a, b are obtained from either the iterative SVD method or the alternating least squares approach as detailed in Reference [22].

When R can be expressed as a sum of p Kronecker products with each product involving two terms, $R = \sum_{i=1}^{p} G_i \otimes F_i$, then the rearrangement and vectorizing operations give $\tilde{R} = \sum_{i=1}^{p} g_i f_i^{\mathrm{T}}$, where $g_i = \text{vec}(G_i)$ and $f_i = \text{vec}(F_i)$. The most important ramification of the

above statement is that now the optimal A and B matrices are linear combinations of the G_i and F_i matrices.

$$A = \alpha_1 G_1 + \alpha_2 G_2 + \cdots + \alpha_p G_p$$

$$B = \beta_1 F_1 + \beta_2 F_2 + \cdots + \beta_p F_p$$

Theorem 4.1

Let $R = \sum_{i=1}^{p} G_i \otimes F_i$, where G_i is an $m_G \times n_G$ matrix and F_i is an $m_F \times n_F$ matrix. Then the NKP matrices A and B are given by

$$A = \alpha_1 G_1 + \alpha_2 G_2 + \cdots + \alpha_p G_p$$

$$B = \beta_1 F_1 + \beta_2 F_2 + \cdots + \beta_p F_p$$

where A has dimension $m_G \times n_G$ and B has dimension $m_F \times n_F$. The optimal NKP matrices A and B are linear combinations of the input matrices G_i and F_i, respectively.

Proof

This proof is not included in Reference [22]. We provide our version of the proof here. First, we define some useful notation. Let A_1 denote the first column of matrix A. Suppose $\tilde{R}$ is square and non-singular. Owing to the special structure of R, we know that $\tilde{R} = \sum_{i=1}^{p} g_i f_i^{\mathrm{T}}$, where $g_i = \mathrm{vec}(G_i)$ and $f_i = \mathrm{vec}(F_i)$. $\tilde{R}$ has an SVD,

$$\tilde{R} = U \begin{pmatrix} D & 0 \\ 0 & 0 \end{pmatrix} V^{\mathrm{T}}$$

where U is an $m_G n_G \times m_G n_G$ matrix, V is $m_F n_F \times m_F n_F$, and D is an $r \times r$ diagonal matrix containing the singular values of $\tilde{R}$ ($\sigma_1, \sigma_2, \ldots, \sigma_r$) with r being the rank of $\tilde{R}$. For a general $\tilde{R}$, $a = \sigma_1 U_1$, where

$$U_1 = \left[\tilde{R} V \begin{pmatrix} D^{-1} & 0 \\ 0 & 0 \end{pmatrix} \right]_1$$

$$= \frac{1}{\sigma_1} [\tilde{R} V]_1$$

$$= \frac{1}{\sigma_1} \tilde{R} V_1$$

Since $\tilde{R} = \sum_{i=1}^{p} g_i f_i^{\mathrm{T}}$,

$$U_1 = \frac{1}{\sigma_1} \left[\sum_{i=1}^{p} g_i f_i^{\mathrm{T}} \right] V_1$$

$$= \frac{1}{\sigma_1} \left[\sum_{i=1}^{p} g_i f_i^{\mathrm{T}} V_1 \right]$$

Since $a = \sigma_1 U_1$, we obtain $a = \sum_{i=1}^{p} g_i f_i^T V_1$ in this case. With $\alpha_i = f_i^T V_1$ we see that a is a linear combination of the g_is, $a = \sum_{i=1}^{p} \alpha_i g_i$. Reversing the vectorizing operation by turning the vectors a and g_i back into matrices gives the desired result; the matrix A is a linear combination of the input matrices G_i. The unvectorizing operation just undoes what the vectorizing operation does. As an example of the unvectorizing operation, consider the vector d obtained by vectorizing the corresponding matrix $m_D \times n_D$ matrix D. Partition d into n_D blocks, each containing m_D elements. The first m_D elements are the first column of the matrix D. After all the partitions of d are unstacked, the matrix D is available. The proof showing b is a linear combination of the f_is is similar. $\square$

Using this fact that the matrices A and B are linear combinations of the G_is and F_is, the original problem is transformed once again. However, Pitsianis and Van Loan's problem transformation given below only holds when G_i and F_i are diagonal or orthogonally similar to a diagonal matrix.

$$\|R - A \otimes B\|_F^2 = \left\| \sum_{i=1}^{p} G_i \otimes F_i - \left(\sum_{i=1}^{p} \alpha_i G_i \right) \otimes \left(\sum_{i=1}^{p} \beta_i F_i \right) \right\|_F^2$$

$$= \sum_{i=1}^{n_G} \sum_{j=1}^{n_F} \left[\left(\sum_{k=1}^{p} \sigma_{G_k}(i) \sigma_{F_k}(j) \right) - \left(\sum_{k=1}^{p} \alpha_k \sigma_{G_k}(i) \right) \left(\sum_{k=1}^{p} \beta_k \sigma_{F_k}(i) \right) \right]^2$$

In the above equation, n_G is the order of G_i, n_F is the order of F_i and $\sigma_{G_k}(i)$ is singular value i of the matrix G_k. The last equality comes from the fact that the square of the Frobenius norm of a matrix is equal to the sum of squares of the singular values of the matrix. Since the R has such a special structure (the sum of Kronecker products of orthogonally diagonalizable matrices), the singular values can be written in terms of the singular values of the much smaller matrices, G_i and F_i. Pitsianis and Van Loan choose $\alpha_1, \alpha_2, \ldots, \alpha_p, \beta_1, \beta_2, \ldots, \beta_p$ so that the above nonlinear function is minimized. They report favourable results with this nonlinear transformation for finding the optimal A, B matrices and using them for a preconditioner. This nearest Kronecker product preconditioner beats the unpreconditioned methods and requires little computational effort on the same example they tested. For the SAN problem the G_i and F_i, in general, are not orthogonally similar to a diagonal matrix, so this problem transformation is not helpful. However, we have discovered the much more general and useful problem transformation provided below:

$$\|R - A \otimes B\|_F^2 = \left\| \sum_{i=1}^{p} G_i \otimes F_i - \left(\sum_{i=1}^{p} \alpha_i G_i \right) \otimes \left(\sum_{i=1}^{p} \beta_i F_i \right) \right\|_F^2$$

$$= \left(\sum_{i=1}^{p} \sum_{j=1}^{p} \operatorname{tr}(G_i^T G_j) \operatorname{tr}(F_i^T F_j) \right)$$

$$- 2 \left(\sum_{i=1}^{p} \left[\left(\sum_{j=1}^{p} \alpha_j \operatorname{tr}(G_i^T G_j) \right) \left(\sum_{j=1}^{p} \beta_j \operatorname{tr}(F_i^T F_j) \right) \right] \right)$$

$$+ \left(\sum_{i=1}^{p} \sum_{j=1}^{p} \alpha_i \alpha_j \operatorname{tr}(G_i^T G_j) \right) \left(\sum_{i=1}^{p} \sum_{j=1}^{p} \beta_i \beta_j \operatorname{tr}(F_i^T F_j) \right)$$

This equality comes from the fact that $\|A\|_F^2 = \mathrm{tr}(A^T A)$, where $\mathrm{tr}(A) = \mathrm{trace}(A)$. The proof of this uses several properties of Kronecker products, namely that $\mathrm{tr}(A \otimes B) = \mathrm{tr}(A)\,\mathrm{tr}(B)$ and $(A \otimes B)(C \otimes D) = AC \otimes BD$ [31]. Below we provide the proof for the simplest case $p = 2$. This can easily be extended to $p > 2$ by induction.

Theorem 4.2
Suppose G_1 and G_2 are square matrices of order n_G and F_1 and F_2 are square matrices of order n_F. Then

$$\|G_1 \otimes F_1 + G_2 \otimes F_2 - (\alpha_1 G_1 + \alpha_2 G_2) \otimes (\beta_1 F_1 + \beta_2 F_2)\|_F^2$$
$$= \mathrm{tr}(G_1^T G_1)\,\mathrm{tr}(F_1^T F_1) + \mathrm{tr}(G_1^T G_2)\,\mathrm{tr}(F_1^T F_2)$$
$$+ \mathrm{tr}(G_2^T G_1)\,\mathrm{tr}(F_2^T F_1) + \mathrm{tr}(G_2^T G_2)\,\mathrm{tr}(F_2^T F_2)$$
$$- 2(\alpha_1\,\mathrm{tr}(G_1^T G_1) + \alpha_2\,\mathrm{tr}(G_1^T G_2))(\beta_1\,\mathrm{tr}(F_1^T F_1) + \beta_2\,\mathrm{tr}(F_1^T F_2))$$
$$- 2(\alpha_1\,\mathrm{tr}(G_2^T G_1) + \alpha_2\,\mathrm{tr}(G_2^T G_2))(\beta_1\,\mathrm{tr}(F_2^T F_1) + \beta_2\,\mathrm{tr}(F_2^T F_2))$$
$$+ (\alpha_1\alpha_1\,\mathrm{tr}(G_1^T G_1) + \alpha_1\alpha_2\,\mathrm{tr}(G_1^T G_2) + \alpha_2\alpha_1\,\mathrm{tr}(G_2^T G_1) + \alpha_2\alpha_2\,\mathrm{tr}(G_2^T G_2))$$
$$\times (\beta_1\beta_1\,\mathrm{tr}(F_1^T F_1) + \beta_1\beta_2\,\mathrm{tr}(F_1^T F_2) + \beta_2\beta_1\,\mathrm{tr}(F_2^T F_1) + \beta_2\beta_2\,\mathrm{tr}(F_2^T F_2))$$

Proof

$$\|G_1 \otimes F_1 + G_2 \otimes F_2 - (\alpha_1 G_1 + \alpha_2 G_2) \otimes (\beta_1 F_1 + \beta_2 F_2)\|_F^2$$
$$= \mathrm{tr}[(G_1^T \otimes F_1^T + G_2^T \otimes F_2^T - (\alpha_1 G_1^T + \alpha_2 G_2^T) \otimes (\beta_1 F_1^T + \beta_2 F_2^T))$$
$$\times (G_1 \otimes F_1 + G_2 \otimes F_2 - (\alpha_1 G_1 + \alpha_2 G_2) \otimes (\beta_1 F_1 + \beta_2 F_2))]$$
$$= \mathrm{tr}[(G_1^T \otimes F_1^T + G_2^T \otimes F_2^T)(G_1 \otimes F_1 + G_2 \otimes F_2)]$$
$$- \mathrm{tr}[(G_1^T \otimes F_1^T + G_2^T \otimes F_2^T)((\alpha_1 G_1 + \alpha_2 G_2) \otimes (\beta_1 F_1 + \beta_2 F_2))]$$
$$- \mathrm{tr}[((\alpha_1 G_1^T + \alpha_2 G_2^T) \otimes (\beta_1 F_1^T + \beta_2 F_2^T))(G_1 \otimes F_1 + G_2 \otimes F_2)]$$
$$+ \mathrm{tr}[((\alpha_1 G_1^T + \alpha_2 G_2^T) \otimes (\beta_1 F_1^T + \beta_2 F_2^T))((\alpha_1 G_1 + \alpha_2 G_2) \otimes (\beta_1 F_1 + \beta_2 F_2))]$$
$$= \mathrm{tr}[(G_1^T \otimes F_1^T)(G_1 \otimes F_1)] + \mathrm{tr}[(G_1^T \otimes F_1^T)(G_2 \otimes F_2)]$$
$$+ \mathrm{tr}[(G_2^T \otimes F_2^T)(G_1 \otimes F_1)] + \mathrm{tr}[(G_2^T \otimes F_2^T)(G_2 \otimes F_2)]$$
$$- 2\,\mathrm{tr}[(G_1^T \otimes F_1^T)((\alpha_1 G_1 + \alpha_2 G_2) \otimes (\beta_1 F_1 + \beta_2 F_2))]$$
$$- 2\,\mathrm{tr}[(G_2^T \otimes F_2^T)((\alpha_1 G_1 + \alpha_2 G_2) \otimes (\beta_1 F_1 + \beta_2 F_2))]$$
$$+ \mathrm{tr}[((\alpha_1 G_1^T + \alpha_2 G_2^T)(\alpha_1 G_1 + \alpha_2 G_2)) \otimes ((\beta_1 F_1^T + \beta_2 F_2^T)(\beta_1 F_1 + \beta_2 F_2))]$$
$$= \mathrm{tr}(G_1^T G_1)\,\mathrm{tr}(F_1^T F_1) + \mathrm{tr}(G_1^T G_2)\,\mathrm{tr}(F_1^T F_2)$$
$$+ \mathrm{tr}(G_2^T G_1)\,\mathrm{tr}(F_2^T F_1) + \mathrm{tr}(G_2^T G_2)\,\mathrm{tr}(F_2^T F_2)$$

$$-2(\alpha_1 \operatorname{tr}(G_1^T G_1) + \alpha_2 \operatorname{tr}(G_1^T G_2))(\beta_1 \operatorname{tr}(F_1^T F_1) + \beta_2 \operatorname{tr}(F_1^T F_2))$$

$$-2(\alpha_1 \operatorname{tr}(G_2^T G_1) + \alpha_2 \operatorname{tr}(G_2^T G_2))(\beta_1 \operatorname{tr}(F_2^T F_1) + \beta_2 \operatorname{tr}(F_2^T F_2))$$

$$+(\alpha_1\alpha_1 \operatorname{tr}(G_1^T G_1) + \alpha_1\alpha_2 \operatorname{tr}(G_1^T G_2) + \alpha_2\alpha_1 \operatorname{tr}(G_2^T G_1) + \alpha_2\alpha_2 \operatorname{tr}(G_2^T G_2))$$

$$\times (\beta_1\beta_1 \operatorname{tr}(F_1^T F_1) + \beta_1\beta_2 \operatorname{tr}(F_1^T F_2) + \beta_2\beta_1 \operatorname{tr}(F_2^T F_1) + \beta_2\beta_2 \operatorname{tr}(F_2^T F_2)) \qquad \square$$

This problem transformation may seem complicated and its value questionable. However, it only involves simple matrix operations on the small G_i and F_i matrices. For $1 \leqslant i,\ j \leqslant p$, the trace of each product, $G_i^T G_j$ and $F_i^T F_j$ must be computed. The trace is an efficient operation and can be computed in $O(n)$ time, where n is the size of the matrix. Further, all p^2 traces of $G_i^T G_j$ need not be computed, since $\operatorname{tr}(G_i^T G_j) = \operatorname{tr}(G_j^T G_i)$. Thus, only $p(p+1)/2$ traces need to be computed and stored for all G_i and $p(p+1)/2$ for all F_i, giving a total of $p(p+1)$. Now a nonlinear optimization code can be applied to the above problem to find the $2p$ unknowns, $\alpha_1, \alpha_2, \ldots, \alpha_p, \beta_1, \beta_2, \ldots, \beta_p$. In a subsequent paper, we discuss the nonlinear optimization problem and the computational effort required in solving it in greater detail [23]. In this paper, our purpose is to develop the theoretical basis for the NKP preconditioner and provide a few small examples to give an intuitive understanding of it.

4.1. Small example

The following example illustrates the success of the NKP preconditioner. Given a matrix R, we want to find its nearest Kronecker product in order to form the NKP preconditioner. Suppose $R = \sum_{i=1}^{3} G_i \otimes F_i$, where

$$G_1 = \begin{pmatrix} 0.2582 & 8.6734 & 7.3851 & 2.8985 \\ 9.2097 & 4.1847 & 0.6726 & 4.3567 \\ 7.0079 & 2.3194 & 3.8430 & 3.2343 \\ 1.9009 & 1.5617 & 9.4272 & 8.6374 \end{pmatrix}, \quad F_1 = \begin{pmatrix} 6.4076 & 0.0353 \\ 3.1576 & 2.5790 \end{pmatrix}$$

$$G_2 = \begin{pmatrix} 0.8921 & 0.7216 & 0.7867 & 0.8677 \\ 0.0167 & 0.6730 & 0.6087 & 0.4536 \\ 0.0562 & 0.3465 & 0.0222 & 0.5719 \\ 0.1458 & 0.1722 & 0.4662 & 0.8215 \end{pmatrix}, \quad F_2 = \begin{pmatrix} 0.5706 & 0.5794 \\ 0.5894 & 0.4318 \end{pmatrix}$$

$$G_3 = \begin{pmatrix} 0.2388 & 0.2024 & 0.8808 & 0.8426 \\ 0.2687 & 0.2352 & 0.8311 & 0.0840 \\ 0.7248 & 0.2522 & 0.5441 & 0.6258 \\ 0.3165 & 0.6702 & 0.8560 & 0.3328 \end{pmatrix}, \quad F_3 = \begin{pmatrix} 0.8064 & 0.7243 \\ 0.6048 & 0.2875 \end{pmatrix}$$

Then

$$
R = \begin{pmatrix}
2.3559 & 0.6989 & 56.1511 & 0.8705 & 48.4799 & 1.3541 & 19.7469 & 1.2152 \\
1.4854 & 1.1197 & 27.9350 & 22.7388 & 24.3155 & 19.6392 & 10.1732 & 8.0921 \\
59.2385 & 0.5290 & 27.3877 & 0.7078 & 5.3271 & 0.9784 & 28.2428 & 0.4773 \\
29.2530 & 23.8365 & 13.7526 & 11.1507 & 2.9851 & 2.2364 & 14.0750 & 11.4561 \\
45.5207 & 0.8046 & 15.2627 & 0.4652 & 25.0760 & 0.5424 & 21.5551 & 0.8986 \\
22.5997 & 18.3062 & 7.6804 & 6.2038 & 12.4768 & 10.0772 & 10.9281 & 8.7682 \\
12.5188 & 0.3807 & 10.6455 & 0.6403 & 61.3624 & 1.2225 & 56.0824 & 1.0215 \\
6.2797 & 5.0564 & 5.4381 & 4.2947 & 30.5599 & 24.7604 & 27.9590 & 22.7265
\end{pmatrix}
$$

Since R is the ordinary sum of Kronecker products involving two terms, the optimal $A = \alpha_1 G_1 + \alpha_2 G_2 + \alpha_3 G_3$ and the optimal $B = \beta_1 F_1 + \beta_2 F_2 + \beta_3 F_3$. Thus, the problem of finding the nearest Kronecker product to R reduces to finding the six coefficients: $\alpha_1, \alpha_2, \alpha_3, \beta_1, \beta_2, \beta_3$. We formulate the nonlinear minimization problem as

$$
\min_{\alpha, \beta} \|R - A \otimes B\|_F^2 = \left\| \sum_{i=1}^{3} G_i \otimes F_i - \left(\sum_{i=1}^{3} \alpha_i G_i \right) \otimes \left(\sum_{i=1}^{3} \beta_i F_i \right) \right\|_F^2
$$

$$
= \left(\sum_{i=1}^{3} \sum_{j=1}^{3} \mathrm{tr}(G_i^{\mathrm{T}} G_j)\, \mathrm{tr}(F_i^{\mathrm{T}} F_j) \right)
$$

$$
-2 \left(\sum_{i=1}^{3} \left[\left(\sum_{j=1}^{3} \alpha_j\, \mathrm{tr}(G_i^{\mathrm{T}} G_j) \right) \left(\sum_{j=1}^{3} \beta_j\, \mathrm{tr}(F_i^{\mathrm{T}} F_j) \right) \right] \right)
$$

$$
+ \left(\sum_{i=1}^{3} \sum_{j=1}^{3} \alpha_i \alpha_j\, \mathrm{tr}(G_i^{\mathrm{T}} G_j) \right) \left(\sum_{i=1}^{3} \sum_{j=1}^{3} \beta_i \beta_j\, \mathrm{tr}(F_i^{\mathrm{T}} F_j) \right)
$$

$$
= 31590.01 - 2[(509.25\alpha_1 + 35.34\alpha_2 + 75.04\alpha_3)
$$

$$
\times (57.68\beta_1 + 13.30\beta_2 + 7.84\beta_3)
$$

$$
+ (35.34\alpha_1 + 5.11\alpha_2 + 7.64\alpha_3)(13.30\beta_1 + 4.78\beta_2 + 2.72\beta_3)
$$

$$
+ (75.04\alpha_1 + 7.64\alpha_2 + 20.31\alpha_3)(7.84\beta_1 + 2.72\beta_2 + 1.62\beta_3)]
$$

$$
+ (509.25\alpha_1^2 + 35.34\alpha_1\alpha_2 + 75.04\alpha_1\alpha_3 + 35.34\alpha_2\alpha_1 + 5.11\alpha_2^2
$$

$$
+ 7.64\alpha_2\alpha_3 + 75.04\alpha_3\alpha_1 + 7.64\alpha_3\alpha_2 + 20.31\alpha_3^2)
$$

$$\times (57.68\beta_1^2 + 13.30\beta_1\beta_2 + 7.84\beta_1\beta_3 + 13.30\beta_2\beta_1 + 4.78\beta_2^2$$

$$+ 2.72\beta_2\beta_3 + 7.84\beta_3\beta_1 + 2.72\beta_3\beta_2 + 1.62\beta_3^2)$$

Using the nonlinear optimization software, multilevel coordinate search (MCS) [24] with the termination criterion of stopping after 60 successive iterates are identical, the optimal $\alpha = [0.6871 \ 0.0800 \ 0.0943]^T$ and $\beta = [1.4290 \ 0.1005 \ 0.1066]^T$ were found in a fraction of a second. Therefore, the optimal NKP matrices, A and B, are

$$A = \begin{pmatrix} 0.2713 & 6.0364 & 5.2203 & 2.1404 \\ 6.3547 & 2.9514 & 0.5892 & 3.0377 \\ 4.8881 & 1.6452 & 2.6937 & 2.3271 \\ 1.3476 & 1.1500 & 6.5955 & 6.0319 \end{pmatrix}, \quad B = \begin{pmatrix} 9.3001 & 0.1858 \\ 4.6360 & 3.7596 \end{pmatrix}$$

And

$$A \otimes B = \begin{pmatrix} 2.5228 & 0.0504 & 56.1391 & 1.1214 & 48.5496 & 0.9698 & 19.9061 & 0.3976 \\ 1.2576 & 1.0198 & 27.9851 & 22.6943 & 24.2017 & 19.6263 & 9.9231 & 8.0471 \\ 59.0997 & 1.1806 & 27.4479 & 0.5483 & 5.4795 & 0.1095 & 28.2513 & 0.5643 \\ 29.4609 & 23.8912 & 13.6827 & 11.0959 & 2.7315 & 2.2151 & 14.0832 & 11.4207 \\ 45.4593 & 0.9081 & 15.3001 & 0.3056 & 25.0512 & 0.5004 & 21.6419 & 0.4323 \\ 22.6613 & 18.3770 & 7.6270 & 6.1851 & 12.4879 & 10.1270 & 10.7884 & 8.7488 \\ 12.5332 & 0.2504 & 10.6955 & 0.2137 & 61.3391 & 1.2253 & 56.0975 & 1.1206 \\ 6.2477 & 5.0666 & 5.3316 & 4.3237 & 30.5773 & 24.7965 & 27.9644 & 22.6775 \end{pmatrix}$$

which may be close enough to the original R to provide the basis for an effective preconditioner. We now check how well $M = A^{-1} \otimes B^{-1}$ (the NKP preconditioner) works. We solve the linear system $Rx = b$. Arbitrarily choosing $b = e$ (where e is the vector of all 1's) and using GMRES(5) [4, 25] with the termination criterion that the norm of the residual is less than 10^{-8}, the unpreconditioned system converges in 33 iterations. The NKP preconditioned system with GMRES converges to the solution in two iterations. The NKP preconditioner also compared favourably with other preconditioners. For example, the threshold-based *ILU* preconditioner [4], with a threshold of 0.01, converged in one iteration. But, the NKP preconditioner requires less storage than the *ILU* preconditioner. The NKP preconditioner stores only the $p \times 1$ vectors for α_i and β_i and A^{-1} and B^{-1}. On the other hand, the *ILU* preconditioner stores the approximate L, U factors of R, which are triangular matrices of the same order as R.

The preconditioned matrix MR is equal to

$$MR = \begin{pmatrix} 1.0024 & -0.0048 & -0.0006 & 0.0020 & 0.0015 & -0.0036 & -0.0018 & 0.0109 \\ -0.0088 & 1.0004 & 0.0025 & -0.0015 & -0.0058 & 0.0012 & 0.0094 & -0.0138 \\ -0.0018 & 0.0013 & 1.0004 & -0.0077 & -0.0047 & 0.0202 & -0.0010 & 0.0033 \\ 0.0057 & 0.0040 & -0.0040 & 1.0131 & 0.0216 & -0.0204 & 0.0042 & -0.0025 \\ -0.0030 & 0.0199 & 0.0005 & -0.0002 & 1.0061 & -0.0249 & -0.0046 & 0.0270 \\ 0.0163 & -0.0264 & -0.0017 & -0.0015 & -0.0276 & 1.0242 & 0.0237 & -0.0338 \\ 0.0028 & -0.0186 & -0.0015 & 0.0091 & -0.0057 & 0.0242 & 1.0053 & -0.0345 \\ -0.0151 & 0.0246 & 0.0080 & -0.0114 & 0.0260 & -0.0244 & -0.0287 & 1.0450 \end{pmatrix}$$

The ideal preconditioned matrix is the identity. The NKP preconditioned matrix follows the pattern of the identity matrix with $\|I - MR\|_F = 0.1565$. Rough estimates of the inverse give good results for preconditioners [26] as our results demonstrate. The eigenvalue distribution of the preconditioned system is another factor that explains the quick convergence of the NKP preconditioned system. The eigenvalues of the NKP preconditioned system are clustered about 1, a desirable convergence property for GMRES. In the nonlinear minimization problem, spending more time searching for α_i and β_i results in a more accurate NKP, so $A \otimes B$ is a better approximation to R. And consequently, for well-conditioned problems, $M = A^{-1} \otimes B^{-1}$ is a better preconditioner, causing the iterative method to converge in fewer iterations. To quantify this, let $R = A \otimes B + E$. Let R be nonsingular and assume the entries in E are small enough in magnitude to insure that $\lim_{n \to \infty} (R^{-1}E)^n = 0$, then by the Neumann series [27]

$$(R - E)^{-1} = [R(I - R^{-1}E)]^{-1} = (I - R^{-1}E)^{-1}R^{-1} = \sum_{k=0}^{\infty} [R^{-1}E]^k R^{-1}$$

The first-order approximation is

$$(R - E)^{-1} \approx R^{-1} + R^{-1}ER^{-1}$$

Then the norm of the difference between the ideal preconditioned system and the NKP preconditioned system is

$$\|I - (A \otimes B)^{-1}R\| = \|I - (R - E)^{-1}R\|$$

$$\approx \|I - (R^{-1} + R^{-1}ER^{-1})R\|$$

$$= \|I - (I + R^{-1}E\| = \| - R^{-1}E\|$$

$$\leqslant \|R^{-1}\|\|E\| = \frac{\|E\|}{\|R\|}\|R\|\|R^{-1}\|$$

$$= \frac{\|E\|}{\|R\|}\operatorname{cond}(R)$$

where $\mathrm{cond}(R)$ is the condition number of R. The magnification factor $\mathrm{cond}(R)$ is counter-balanced by $\|E\|$. A smaller $\|E\|$ means the NKP preconditioned system will be closer to the ideal preconditioned system, provided the system is well-conditioned.

This small example provides hope for solving SAN problems, but, we must ask if this NKP work can be extended to the case $R = Q = \sum_{j=1}^{T} \otimes_{i=1}^{N} Q_j^{(i)}$. (The SAN descriptor Q is always in this form.) Q is a sum of T Kronecker products where each product involves N matrices, rather than just two matrices. Since Pitsianis and Van Loan's work hinges on the matrix SVD, which expresses $\tilde{R}$ as $U\Sigma V^{\mathrm{T}}$ and the optimal A and B matrices are derived from the dominant left and right singular vectors, we need something beyond the matrix SVD to extend the NKP to SANs. It is now time to look at the work of de Lathauwer.

5. MULTILINEAR ALGEBRA AND THE HIGHER-ORDER SVD

The SAN NKP problem can be written as

$$\min \|Q - A_1 \otimes A_2 \otimes \cdots \otimes A_N\|_F^2 = \min \left\| \sum_{j=1}^{T} \otimes_{i=1}^{N} Q_j^{(i)} - A_1 \otimes A_2 \otimes \cdots \otimes A_N \right\|_F^2$$

To extend Pitsianis and Van Loan's method for determining $A \otimes B$, we need: (1) a rearrangement $\tilde{Q}$ of Q and (2) a decomposition of $\tilde{Q}$ into something with more terms than just an SVD. Perhaps to extend beyond the 2-optimal matrix case ($A \otimes B$) to the N-optimal matrix case ($A_1 \otimes A_2 \otimes \cdots \otimes A_N$) we need to extend a two-dimensional linear algebra to an N-dimensional algebra. This N-dimensional algebra, multilinear algebra, has been studied extensively by de Lathauwer [28–30].

The fundamental object of multilinear algebra is the tensor. A first-order tensor is a vector. A second-order tensor is a matrix. A third-order tensor can be visually represented as a three-dimensional box. For example, a third-order tensor R of dimension $3 \times 2 \times 4$ has elements labelled (i, j, k) for $i = 1, 2, 3$, $j = 1, 2$, and $k = 1, 2, 3, 4$ and is shown in Figure 1. Nth-order tensors may be too difficult to visually represent but they exist with each element labelled similarly, as an N-dimensional vector.

Just as linear algebra operates on matrices and vectors, multilinear algebra operates on tensors. Multilinear algebra becomes deep very quickly. For more information, see References [28–30]. Since the material is dense, we extract only what is needed for the SAN NKP preconditioner. Operations include inner and outer products, scalar products and multiplication of a tensor by a matrix or vector. For example, the outer product of N vectors results in an Nth-order tensor. This is analogous to the matrix outer product of two vectors, which results in a second-order tensor, a matrix. Tensors also have rank and an SVD. For the matrix case, a rank-p matrix can be expressed as sums of outer products of two vectors using the SVD. The SVD of a tensor is called the higher-order SVD (HOSVD). A rank-1 Nth-order tensor can be expressed as an outer product of N vectors. de Lathauwer uses the HOSVD of an Nth-order tensor R of dimension $I_1 \times I_2 \times \cdots \times I_N$ to express a tensor R as a sum of rank-1 tensors,

$$R = \sum_{i_1=1}^{I_1} \sum_{i_2=1}^{I_2} \cdots \sum_{i_N=1}^{I_N} s_{i_1, i_2, \ldots, i_N} u_{i_1}^{(1)} \circ u_{i_2}^{(2)} \circ \cdots \circ u_{i_N}^{(N)}$$
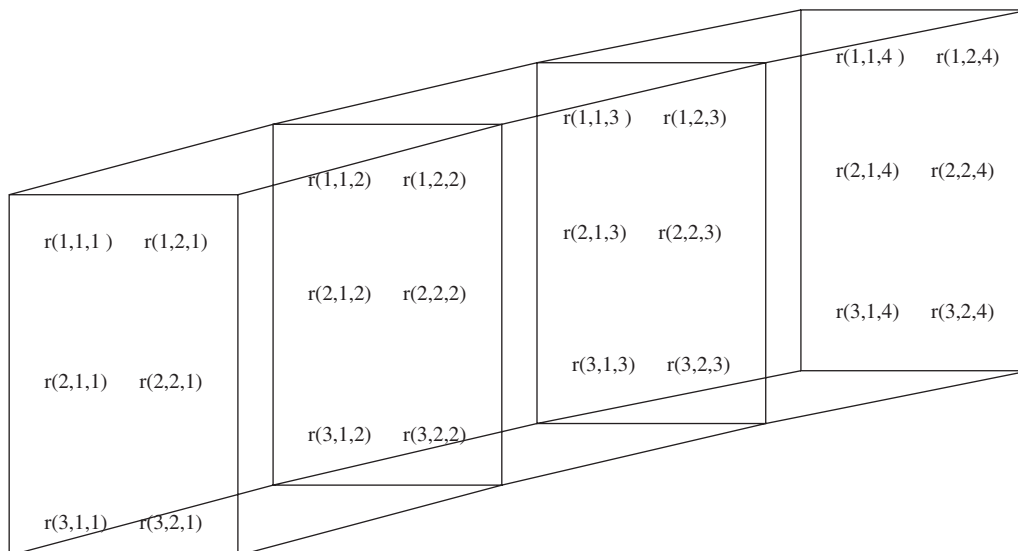
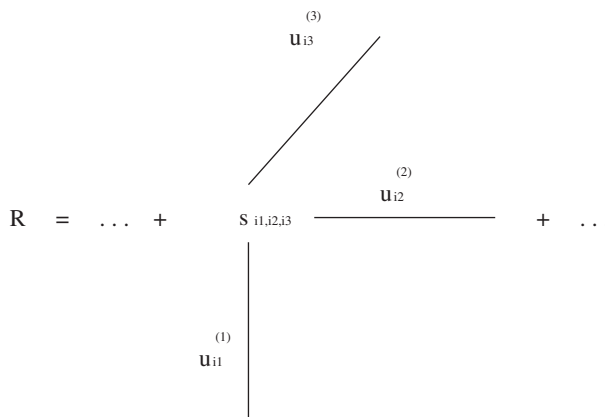Figure 1. Third-order tensor R of dimension $3 \times 2 \times 4$.

Figure 2. Visualization of HOSVD of third-order tensor R.

where $u_{i_j}^{(j)}$ is an all-orthogonal [28] $I_j \times 1$ vector for $j = 1, \ldots, N$, the $s_{i_1, i_2, \ldots, i_N}$ are higher-order singular values, which need not be positive and $\circ$ represents the outer product operation. A matrix SVD of a rank-p matrix R can be written as $R = \sum_{i=1}^{p} \sigma_i u_i v_i^{\mathrm{T}}$. With the HOSVD, a third-order tensor R of dimension $I_1 \times I_2 \times I_3$ can be expressed as $\sum_{i_1=1}^{I_1} \sum_{i_2=1}^{I_2} \sum_{i_3=1}^{I_3} s_{i_1, i_2, i_3} u_{i_1}^{(1)} \circ u_{i_2}^{(2)} \circ u_{i_3}^{(3)}$. See Figure 2, which is taken from Reference [28]. The lines represent the singular vectors in three-dimensional space.

5.1. Finding higher-order NKP for a general matrix

We use the HOSVD to find a SAN NKP preconditioner. In the first step, we extend Pitsianis and Van Loan's work to solve the problem, $\min \|R - A_1 \otimes A_2 \otimes \cdots \otimes A_N\|_F^2$, for a general matrix R. We would like to approximate some rearrangement $\tilde{R}$ of R by an outer product of N vectors $(a_1 \circ a_2 \circ \cdots \circ a_N)$. In the end, we hope that reversing the vectorizing operation will give us the optimal approximation matrices $A_1, A_2, \ldots, AN$. Knowledge of multilinear algebra reveals that the rearrangement $\tilde{R}$ of R should be an Nth-order tensor since the outer product of N vectors is an Nth-order tensor. Thus, just as in the matrix case $A \otimes B$, we define a rearrangement operation so that

$$\min \|R - A_1 \otimes A_2 \otimes \cdots \otimes A_N\|_F^2 = \min \|\tilde{R} - a_1 \circ a_2 \circ \cdots \circ a_N\|_F^2$$

where $a_1 = \text{vec}(A_1)$, $a_2 = \text{vec}(A_2), \ldots, a_N = \text{vec}(A_N)$, $\tilde{R}$ is an Nth-order tensor and $a_1 a_2 \circ \cdots \circ a_N$ is a rank-1 Nth-order tensor.

First, we define our rearrangement operation. Then, we prove that with this definition the sums of squares in the above equation match. In the most general case, we want to approximate the $m_1 m_2 \cdots m_N \times n_1 n_2 \cdots n_N$ matrix R by $A_1 \otimes A_2 \otimes \cdots \otimes A_N$, where A_1 is $m_1 \times n_1$, A_2 is $m_2 \times n_2$ and A_N is $m_N \times n_N$. The matrix R consists of $m_1 n_1$ blocks, each of size $m_2 \cdots m_N \times n_2 \cdots n_N$. We label the (i, j) block of R as $R_{i,j}$. Define a rearrangement operation, $\Gamma_N(R)$, which, when applied to a matrix R of dimension $m_1 m_2 \cdots m_N \times n_1 n_2 \cdots n_N$ results in an Nth-order tensor $\tilde{R}$ of dimension $m_1 n_1 \times m_2 n_2 \times \cdots \times m_N n_N$. Define $\Gamma_N(R)$ recursively as follows: vectorize the $m_1 n_1$ blocks of the matrix R, and then apply $\Gamma_{N-1}(R_{i,j})$ to each block. The scalars m_i and n_i are the user-defined dimensions of the NKP matrix A_i. The only restriction is that the dimension of the Kronecker product of the NKP matrices $A_1, A_2, \ldots A_N$ match that of R. Pitsianis and Van Loan defined $\Gamma_2(R)$ in Reference [22]. $\Gamma_2(R)$ takes the matrix R and transforms it into a 2nd-order tensor so that the sum of squares of $\|R - A \otimes B\|$ are identical to those of $\|\tilde{R} - a \circ b\|$. We now describe $\Gamma_3(R)$ with an example. Suppose we approximate $R \in \mathfrak{R}^{12 \times 12}$ with $A \otimes B \otimes C$, where A is 2×2, B is 2×2 and C is 3×3. The dimensions of A, B and C are arbitrarily chosen with the lone restriction that these dimensions are conformable with the dimension of R. R is then composed of four blocks, each of size 6×6. To distinguish between matrix blocks and matrix elements we use the following notation: matrix blocks are capitalized with subscripted indices while matrix elements are lowercase with indices in parentheses.

$$R = \left(\begin{array}{c|c} R_{1,1} & R_{1,2} \\ \hline R_{2,1} & R_{2,2} \end{array} \right)$$

$$= \left(\begin{array}{cccc|cccc} r(1,1) & r(1,2) & \cdots & r(1,6) & r(1,7) & r(1,8) & \cdots & r(1,12) \\ \vdots & \ddots & & \vdots & \vdots & \ddots & & \vdots \\ r(6,1) & r(6,2) & \cdots & r(6,6) & r(6,7) & r(6,8) & \cdots & r(6,12) \\ \hline r(7,1) & r(7,2) & \cdots & r(7,6) & r(7,7) & r(7,8) & \cdots & r(7,12) \\ \vdots & \ddots & & \vdots & \vdots & \ddots & & \vdots \\ r(12,1) & r(12,2) & \cdots & r(12,6) & r(12,7) & r(12,8) & \cdots & r(12,12) \end{array} \right)$$
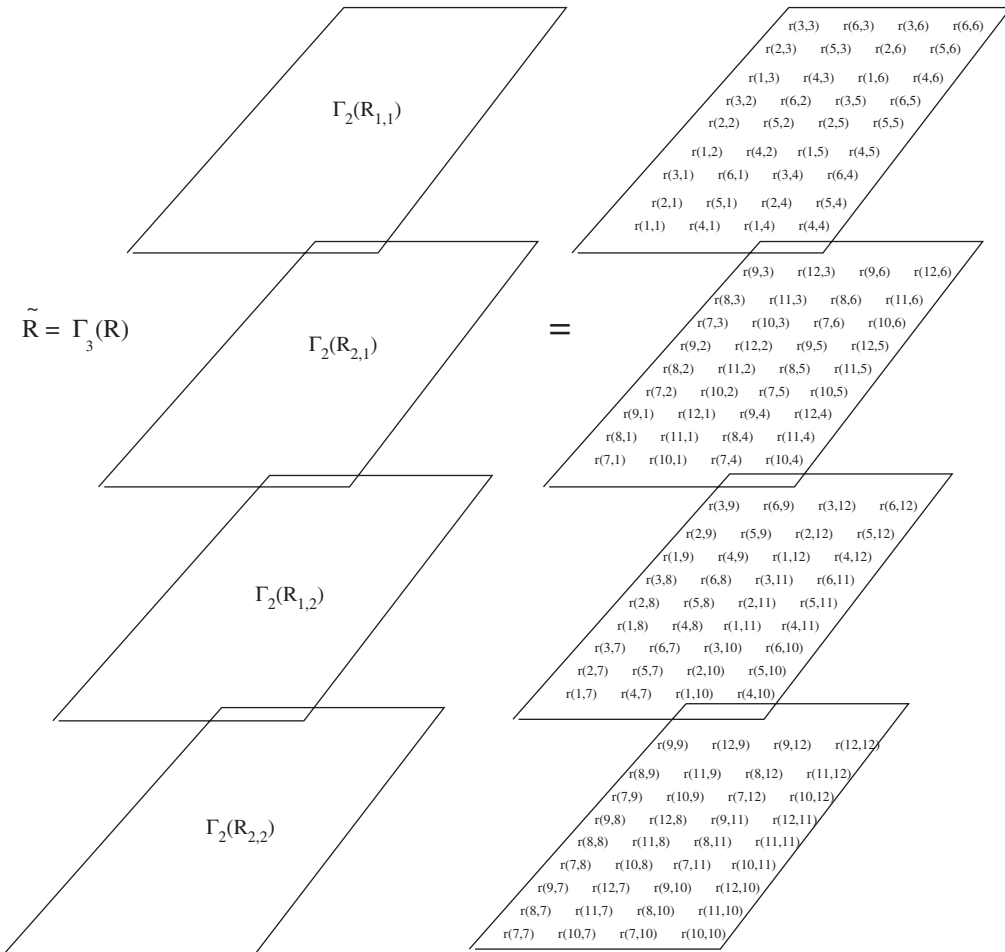
Figure 3. Visualization of third-order tensor $\tilde{R}$ of dimension $4 \times 4 \times 9$.

The rearrangement operation transforms the matrix R into a third-order tensor $\tilde{R} = \Gamma_3(R)$. $\Gamma_3(R)$ is a third-order tensor of dimension $4 \times 4 \times 9$ and is represented in Figure 3.

We now prove that the higher-order rearrangement operation accomplishes the desired transformation.

Theorem 5.1
A_1 is an $m_1 \times n_1$ matrix, A_2 is an $m_2 \times n_2$ matrix and A_N is an $m_N \times n_N$ matrix. R is an $m_1 m_2 \ldots m_N \times n_1 n_2 \ldots n_N$ matrix. $\Gamma_N(R)$ is the rearrangement of the matrix R into an Nth-order tensor of dimension $m_1 n_1 \times m_2 n_2 \times \ldots \times m_N n_N$. Then $\|R - A_1 \otimes A_2 \otimes \cdots \otimes A_N\|_F^2 = \|\Gamma_N(R) - a_1 \circ a_2 \circ \cdots \circ a_N\|_F^2$, where $a_1 = \text{vec}(A_1)$, $a_2 = \text{vec}(A_2)$, ..., $a_N = \text{vec}(A_N)$.

Proof
We prove that $\|R - A_1 \otimes A_2 \otimes \cdots \otimes A_N\|_F^2 = \|\Gamma_N(R) - a_1 \circ a_2 \circ \cdots \circ a_N\|_F^2$ by induction. This proof uses the fact that an Nth-order tensor consists of several $N-1$st-order tensors. For the

base case, we show that $\|R - A_1 \otimes A_2 \otimes A_3\|_F^2 = \|\Gamma_3(R) - a_1 \circ a_2 \circ a_3\|_F^2$.

$$\|R - A_1 \otimes A_2 \otimes A_3\|_F^2 = \sum_{i=1}^{m_1} \sum_{j=1}^{n_1} \|R_{i,j} - (A_1 \otimes A_2 \otimes A_3)_{i,j}\|_F^2$$

$$= \sum_{i=1}^{m_1} \sum_{j=1}^{n_1} \|R_{i,j} - A_{1_{i,j}}(A_2 \otimes A_3)\|_F^2$$

Now we can use the result of Pitsianis and Van Loan which says that $\|R - A_1 \otimes A_2\|_F^2 = \|\Gamma_2(R) - a_1 \circ a_2\|_F^2$.

$$\|R - A_1 \otimes A_2 \otimes A_3\|_F^2 = \sum_{i=1}^{m_1} \sum_{j=1}^{n_1} \|\Gamma_2(R_{i,j}) - A_{1_{i,j}}(a_2 \circ a_3)\|_F^2$$

$$= \sum_{k=1}^{m_1 n_1} \|R_k - a_{1_k}(a_2 \circ a_3)\|_F^2$$

$$= \|\Gamma_3(R) - a_1 \circ a_2 \circ a_3\|_F^2$$

where a_{1_k} is the kth element of the $m_1 n_1 \times 1$ vector obtained by the operation vec(A_1). Given that a_{1_k} corresponds to the matrix element $A_{1_{i,j}}$, then R_k is defined similarly as the rearrangement of the block matrix $R_{i,j}$. The final equality comes from the fact that a third-order tensor $\Gamma_3(R)$ is actually $m_1 n_1$ second-order tensors and thus the sum of squares of the third-order tensor is the same as the sum of the $m_1 n_1$ sums of squares of the second-order tensors.

For the induction step, we assume $\|R - A_1 \otimes A_2 \otimes \cdots \otimes A_N\|_F^2 = \|\Gamma_N(R) - a_1 \circ a_2 \circ \cdots \circ a_N\|_F^2$ and show that $\|R - A_1 \otimes A_2 \otimes \cdots \otimes A_N \otimes A_{N+1}\|_F^2 = \|\Gamma_{N+1}(R) - a_1 \circ a_2 \circ \cdots \circ a_N \circ a_{N+1}\|_F^2$.

$$\|R - A_1 \otimes A_2 \otimes \cdots \otimes A_N \otimes A_{N+1}\|_F^2 = \sum_{i=1}^{m_1} \sum_{j=1}^{n_1} \|R_{i,j} - (A_1 \otimes A_2 \otimes \cdots \otimes A_{N+1})_{i,j}\|_F^2$$

$$= \sum_{i=1}^{m_1} \sum_{j=1}^{n_1} \|R_{i,j} - A_{1_{i,j}}(A_2 \otimes A_3 \otimes \cdots \otimes A_{N+1})\|_F^2$$

$$= \sum_{i=1}^{m_1} \sum_{j=1}^{n_1} \|\Gamma_N(R_{i,j}) - A_{1_{i,j}}(a_2 \circ a_3 \circ \cdots \circ a_{N+1})\|_F^2$$

$$= \sum_{k=1}^{m_1 n_1} \|R_k - a_{1_k}(a_2 \circ a_3 \circ \cdots \circ a_{N+1})\|_F^2$$

$$= \|\Gamma_{N+1}(R) - a_1 \circ a_2 \circ \cdots \circ a_{N+1}\|_F^2 \qquad \square$$

In the two-dimensional matrix case we approximate the matrix $\tilde{R}$ by the outer product of two vectors $(a \circ b = ab^\mathrm{T})$. The truncated SVD provided the optimal a, b vectors for the rank-1 approximation problem. Unfortunately, we now discover an important difference between multilinear algebra and linear algebra. de Lathauwer proves that, unlike the matrix case, the

truncated HOSVD does not provide the best rank-1 approximation of an Nth-order tensor. He provides a complicated algorithm (which requires the formation of the Nth-order tensor $\tilde{R}$ and a tensor-matrix multiplication algorithm) in Reference [30] that does find the best rank-1 approximation of an Nth-order tensor. However, de Lathauwer also states that, while the truncated HOSVD does not provide the best rank-1 approximation, it does generally provide a *good* rank-1 approximation. Thus, using the truncated HOSVD we have

$$a_1 \approx s_{1,1,\dots,1} u_1^{(1)}$$

$$a_2 \approx u_1^{(2)}$$

$$\vdots$$

$$a_N \approx u_1^{(N)}$$

where $s_{1,1,\dots,1}$ is the first pseudosingular value of $\tilde{R}$, and $u_1^{(j)}$ are the first singular vectors of $\tilde{R}$. Then, $a_1 \circ a_2 \circ \cdots \circ a_N$ hopefully provides a good rank-1 approximation of the Nth-order tensor $\tilde{R}$. And reversing the vectorizing operation, $A_1 \otimes A_2 \otimes \cdots \otimes A_N$ provides a good approximation of the original matrix R. Finally, we have the desired result. Although this result is not as neat and exact as Pitsianis and Van Loan's optimal two-matrix case, it is a nice starting approximation. Ultimately, we are only interested in finding a suitable *approximate* preconditioner for SANs.

5.2. Finding higher-order NKP for a SAN matrix

This is all very promising news theoretically, but practically we would rather not form or store the Nth-order tensor $\tilde{R}$. Thus, we now find the optimal $A_1, A_2, \dots, A_N$ matrices without forming $\tilde{R}$, or finding the HOSVD of $\tilde{R}$. We extend Pitsianis and Van Loan's work on matrices with special structure. The goal is solving the problem min $\|Q - A_1 \otimes A_2 \otimes \cdots \otimes A_N\|_F^2$, where Q is the SAN descriptor with the special structure, $\sum_{j=1}^{T} \otimes_{i=1}^{N} Q_j^{(i)}$. In the optimal two-matrix case when $R = \sum_{j=1}^{T} \otimes_{i=1}^{\mathbf{2}} Q_j^{(i)}$, the optimal A and B matrices are linear combinations of the $Q_j^{(i)}$ matrices, $A = \alpha_1 Q_1^{(1)} + \alpha_2 Q_2^{(1)} + \cdots + \alpha_T Q_T^{(1)}$ and $B = \beta_1 Q_1^{(2)} + \beta_2 Q_2^{(2)} + \cdots + \beta_T Q_T^{(2)}$. Does this fact extend to $A_1, A_2, \dots, A_N$? Is $A_1 = \alpha_1 Q_1^{(1)} + \alpha_2 Q_2^{(1)} + \cdots + \alpha_T Q_T^{(1)}$, $A_2 = \beta_1 Q_1^{(2)} + \beta_2 Q_2^{(2)} + \cdots + \beta_T Q_T^{(2)}, \dots$, and $A_N = \eta_1 Q_1^{(N)} + \eta_2 Q_2^{(N)} + \cdots + \eta_T Q_T^{(N)}$? The answer to this question is 'Yes, approximately'.

Theorem 5.2
Let $Q = \sum_{j=1}^{T} \otimes_{i=1}^{N} Q_j^{(i)}$ be the SAN descriptor. Then the approximate NKP matrices A_1, $A_2, \dots, A_N$ such that $Q \approx A_1 \otimes A_2 \otimes \cdots \otimes A_N$ follow: $A_1 \approx \alpha_1 Q_1^{(1)} + \alpha_2 Q_2^{(1)} + \cdots + \alpha_T Q_T^{(1)}$, $A_2 \approx \beta_1 Q_1^{(2)} + \beta_2 Q_2^{(2)} + \cdots + \beta_T Q_T^{(2)}, \dots$, and $A_N \approx \eta_1 Q_1^{(N)} + \eta_2 Q_2^{(N)} + \cdots + \eta_T Q_T^{(N)}$.

Proof
de Lathauwer has proven that every Nth-order tensor $\tilde{R}$ of dimension $I_1 \times I_2 \times \cdots \times I_N$ has a HOSVD:

$$\tilde{R} = S \times_1 U^{(1)} \times_2 U^{(2)} \times_3 \cdots \times_N U^{(N)}$$

where $U^{(n)} = [U_1^{(1)} U_2^{(2)} \cdots U_{I_N}^{(n)}]$ is a unitary $(I_N \times I_N)$ matrix, S is an Nth-order tensor of dimension $I_1 \times I_2 \times \cdots \times I_N$ and $\times_j$ represents tensor-matrix multiplication along dimension j as detailed in Reference [28]. Using the unfolding operation in Reference [28], we 'unfold' the Nth-order tensors into matrices. An Nth-order tensor $\tilde{R}$ can be unfolded along any of its N dimensions. Without loss of generality, suppose $\tilde{R}$ is unfolded along the $P(1)$ dimension and is represented in matrix terms as

$$\tilde{R}_{P(1)} = U^{(P(1))} S[U^{(P(2))} \otimes \cdots \otimes U^{(P(N))}]^{\mathrm{T}}$$

where the unfolded matrix $\tilde{R}_{P(1)}$ has dimension $I_{P(1)} \times I_{P(2)} \cdots I_{P(N)}$, S has dimension $I_{P(1)} \times I_{P(2)} \cdots I_{P(N)}$ and $P(1, 2, \ldots, N)$ is a particular permutation of $(1, 2, \ldots, N)$ with $P(i)$ representing the ith element of that permutation. Then $\tilde{R}_{P(1)}$ has a matrix SVD [28] given by

$$\tilde{R}_{P(1)} = U^{P(1)} \Sigma^{P(1)} V^{P(1)T}$$

where $\Sigma^{P(1)}$ has dimension $I_{P(1)} \times I_{P(1)}$ and is a diagonal matrix containing the singular values of the matrix $\tilde{R}_{P(1)}$, $V^{P(1)}$ which has dimension $I_{P(2)} \ldots I_{P(N)} \times I_{P(1)}$ is an orthogonal matrix defined by $V^{P(1)T} = \tilde{S}_{P(1)}[U^{P(2)} \otimes \cdots \otimes U^{P(N)}]^{\mathrm{T}}$. $\tilde{S}_{P(1)}$ is a normalized version of the tensor S unfolded to the matrix $S_{P(1)}$. Specifically, $S_{P(1)} = \Sigma^{P(1)} \tilde{S}_{P(1)}$ [28]. Since $V^{P(1)}$ is orthogonal and $\Sigma^{P(1)}$ is a square diagonal matrix, we can isolate $U_1^{P(1)}$, the first column of

$$U_1^{P(1)} = [\tilde{R}_{P(1)} V^{P(1)} (\Sigma^{P(1)})^{-1}]_1$$

$$= \frac{1}{s_{1,1,\ldots,1}} \tilde{R}_{P(1)} V_1^{P(1)}$$

where $1/s_{1,1,\ldots,1}$ is the $(1,1)$ – element of the matrix $\Sigma^{P(1)}$. Without loss of generality, assume $P(1) = 1$. Thus,

$$s_{1,1,\ldots,1} U_1^{(1)} = \tilde{R}_{(1)} V_1^{(1)}$$

Now we use $\tilde{R}$'s special structure. The original matrix $R = \sum_{j=1}^{T} \otimes_{i=1}^{N} Q_j^{(i)}$. Thus, $\tilde{R} = \sum_{j=1}^{T} q_j^{(1)} \circ q_j^{(2)} \circ \cdots \circ q_j^{(N)}$. Unfolding the tensor $\tilde{R}$ into a matrix gives

$$\tilde{R} = \sum_{j=1}^{T} q_j^{(1)} (q_j^{(2)} \otimes \cdots \otimes q_j^{(N)})^{\mathrm{T}}$$

Hence,

$$s_{1,1,\ldots,1} U_1^{(1)} = \sum_{j=1}^{T} q_j^{(1)} (q_j^{(2)} \otimes \cdots \otimes q_j^{(N)})^{\mathrm{T}} [U^{(P(2))} \otimes \cdots \otimes U^{(P(N))}]_1$$

Let the scalar product $(q_j^{(2)} \otimes \cdots \otimes q_j^{(N)})^{\mathrm{T}} [U^{(P(2))} \otimes \cdots \otimes U^{(P(N))}]_1 = \alpha_j$. Then $s_{1,1,\ldots,1} U_1^{(1)} = \sum_{j=1}^{T} \alpha_1 q_j^{(1)}$. Since $a_1 \approx s_{1,1,\ldots,1} U_1^{(1)}$, then a_1 is approximately a linear combination of the $q_j^{(1)}$'s. Reversing the vectorizing operation gives the desired result; the matrix A_1 is approximately a linear combination of the input matrices $Q_j^{(1)}$'s. The proof showing a_2 (a_N) is approximately a linear combination of the $q_j^{(2)}$'s ($q_j^{(N)}$'s) is similar. □

Similar to the two-matrix case, the original N-matrix problem is transformed using the trace definition of the squared Frobenius norm:

$$\|Q - A_1 \otimes A_2 \otimes \cdots \otimes A_N\|_F^2$$

$$\approx \left\| \sum_{j=1}^{T} \otimes_{i=1}^{N} Q_j^{(i)} - \left(\sum_{j=1}^{T} \alpha_j Q_j^{(1)} \right) \otimes \left(\sum_{j=1}^{T} \beta_j Q_j^{(2)} \right) \otimes \cdots \otimes \left(\sum_{j=1}^{T} \eta_j Q_j^{(N)} \right) \right\|_F^2$$

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

$$+ \prod_{k=1}^{N} \left[\sum_{i=1}^{T} \sum_{j=1}^{T} \alpha_i^{(k)} \alpha_j^{(k)} \mathrm{tr}(Q_i^{(k)T} Q_j^{(k)}) \right] \tag{3}$$

In the above equation, $\alpha_i^{(1)} = \alpha_i$, $\alpha_i^{(2)} = \beta_i$ and $\alpha_i^{(N)} = \eta_i$ for ease of notation. The unknowns $\alpha_1, \alpha_2, \ldots, \alpha_T, \beta_1, \beta_2, \ldots, \beta_T, \ldots, \eta_1, \eta_2, \ldots, \eta_T$ (that is, all the $\alpha_i^{(k)}s$) are chosen so that the above nonlinear function is minimized. This nonlinear function of NT variables 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 problem transformation becomes impractical.

This drawback relating to the size of the nonlinear optimization problem can be circumvented with the results of Plateau and her co-workers. In practical modelling 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. Reference [18] shows 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 and/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 Markov chains of size 10^{10}. Thus, it is reasonable to expect that after grouping techniques are used, the nonlinear optimization problem in Equation (3) contains less than 80 unknowns.

5.2.1. Small example A. The remainder of this section deals with applications of the NKP preconditioner to small artificial examples. First, we apply the NKP preconditioner to a nonsingular system with coefficient matrix $Q = \sum_{j=1}^{T} \otimes_{i=1}^{3} Q_j^{(i)}$. Then, we apply the NKP preconditioner to a singular system arising from a small SAN.

For $Q = \sum_{j=1}^{T} \otimes_{i=1}^{N} Q_j^{(i)}$, we choose $N = 3$ and $T = 3$. $Q = Q_1^{(1)} \otimes Q_1^{(2)} \otimes Q_1^{(3)} + Q_2^{(1)} \otimes Q_2^{(2)} \otimes Q_2^{(3)} + Q_3^{(1)} \otimes Q_3^{(2)} \otimes Q_3^{(3)}$. The matrix Q has the SAN structure by design but since the $Q_j^{(i)}$ matrices for $i, j = 1, 2, 3$ are generated randomly, Q is not a SAN. In fact, Q is nonsingular.

Let

$$Q_1^{(1)} = \begin{pmatrix} 0.1549 & 0.2041 & 0.4310 & 0.7391 \\ 0.5258 & 0.5108 & 0.4035 & 0.6140 \\ 0.2047 & 0.3916 & 0.3058 & 0.9406 \\ 0.1405 & 0.9370 & 0.6563 & 0.1824 \end{pmatrix}$$

$$Q_2^{(1)} = \begin{pmatrix} 3.5242 & 3.2986 & 7.2004 & 2.0621 \\ 0.9998 & 0.3773 & 6.5241 & 7.2157 \\ 3.3705 & 0.7920 & 1.3297 & 4.3136 \\ 8.7454 & 0.3342 & 7.9736 & 9.2875 \end{pmatrix}$$

$$Q_3^{(1)} = \begin{pmatrix} 0.4327 & 0.2142 & 0.2849 & 0.0558 \\ 0.6124 & 0.4113 & 0.9867 & 0.5043 \\ 0.3487 & 0.5132 & 0.2643 & 0.7546 \\ 0.8989 & 0.0215 & 0.1360 & 0.9965 \end{pmatrix}$$

$$Q_1^{(2)} = \begin{pmatrix} 1.6892 & 1.0948 & 0.9524 \\ 1.7891 & 0.8469 & 1.1813 \\ 0.0423 & 1.0725 & 1.6771 \end{pmatrix}, \quad Q_1^{(3)} = \begin{pmatrix} 0.0590 & 0.6475 \\ 0.7470 & 0.1842 \end{pmatrix}$$

$$Q_2^{(2)} = \begin{pmatrix} 0.6287 & 0.1407 & 0.7140 \\ 0.3458 & 0.9492 & 0.9294 \\ 0.7252 & 0.5216 & 0.0723 \end{pmatrix}, \quad Q_2^{(3)} = \begin{pmatrix} 1.5063 & 1.0292 \\ 0.0452 & 2.6041 \end{pmatrix}$$

$$Q_3^{(2)} = \begin{pmatrix} 2.7855 & 3.1636 & 4.9718 \\ 1.2599 & 0.4099 & 1.3927 \\ 0.6731 & 3.3888 & 1.5116 \end{pmatrix}, \quad Q_3^{(3)} = \begin{pmatrix} 0.2794 & 0.0892 \\ 0.3283 & 0.5172 \end{pmatrix}$$

Since Q is order 24, we only show the upper left block of Q.

$$Q(1:7, 1:7) = \begin{pmatrix} 3.6900 & 2.5575 & 1.1395 & 0.7423 & 4.4001 & 2.8771 & 3.3112 \\ 0.6914 & 6.4420 & 0.5986 & 2.0307 & 0.9303 & 7.6925 & 0.5472 \\ 2.0044 & 1.4823 & 5.0960 & 3.5434 & 5.1132 & 3.5434 & 1.8152 \\ 0.4411 & 3.5066 & 0.3075 & 8.8268 & 0.4827 & 8.8753 & 0.4129 \\ 3.9318 & 2.6607 & 3.1883 & 2.1301 & 0.5820 & 0.4889 & 3.6444 \\ 0.2161 & 6.8078 & 0.6887 & 5.5756 & 0.4204 & 1.0499 & 0.1620 \\ 1.4760 & 1.3742 & 0.7873 & 0.6904 & 1.9557 & 1.3306 & 0.7284 \end{pmatrix}$$

To find the NKP $A_1 \otimes A_2 \otimes A_3$ of Q, we use the nonlinear minimization problem of equation (3). Using the nonlinear optimization software MCS [24], in a fraction of a second we find $\alpha = [0.2651 \; 1.304 \; 0.7417]^{\mathrm{T}}$, $\beta = [0.0180 \; 2.000 \; 0.0346]^{\mathrm{T}}$ and $\gamma = [0.0326 \; 0.3379 \; 0.0922]^{\mathrm{T}}$, where α is the vector of linear coefficients for A_1, β for A_2 and γ for A_3. The algorithm terminated after stagnation for 60 successive iterations. Since $A_1 \approx \alpha_1 Q_1^{(1)} + \alpha_2 Q_2^{(1)} + \alpha_3 Q_3^{(1)}$, $A_2 \approx \beta_1 Q_1^{(2)} + \beta_2 Q_2^{(2)} + \beta_3 Q_3^{(2)}$ and $A_3 \approx \gamma_1 Q_1^{(3)} + \gamma_2 Q_2^{(3)} + \gamma_3 Q_3^{(3)}$, we replace the approximation sign with an equal sign and form

$$
A_1 = \begin{pmatrix} 4.9593 & 4.5161 & 9.7184 & 2.9273 \\ 1.8979 & 0.9326 & 9.3496 & 9.9497 \\ 4.7098 & 1.5176 & 2.0117 & 6.4362 \\ 12.1123 & 0.7003 & 10.6764 & 12.9030 \end{pmatrix}
$$

$$
A_2 = \begin{pmatrix} 1.3843 & 0.4106 & 1.6171 \\ 0.7674 & 1.9277 & 1.9283 \\ 1.4745 & 1.1797 & 0.2271 \end{pmatrix}, \quad A_3 = \begin{pmatrix} 0.5366 & 0.3771 \\ 0.0699 & 0.9336 \end{pmatrix}
$$

Now compare the top left block of the NKP with the original Q matrix.

$$
(A_1 \otimes A_2 \otimes A_3)(1:7, 1:7) = \begin{pmatrix} 3.6841 & 2.5885 & 1.0927 & 0.7678 & 4.3038 & 3.0240 & 3.3548 \\ 0.4798 & 6.4090 & 0.1423 & 1.9010 & 0.5605 & 7.4871 & 0.4369 \\ 2.0423 & 1.4350 & 5.1304 & 3.6048 & 5.1320 & 3.6059 & 1.8597 \\ 0.2660 & 3.5528 & 0.6681 & 8.9251 & 0.6683 & 8.9278 & 0.2422 \\ 3.9243 & 2.7573 & 3.1395 & 2.2059 & 0.6045 & 0.4247 & 3.5735 \\ 0.5111 & 6.8268 & 0.4089 & 5.4616 & 0.0787 & 1.0516 & 0.4654 \\ 1.4098 & 0.9906 & 0.4182 & 0.2938 & 1.6470 & 1.1572 & 0.6928 \end{pmatrix}
$$

The NKP is not as close to Q as in the previous two-dimensional example from Section 7.2.1 but, nevertheless, the NKP preconditioned iteration matrix MQ (where $M = A_1^{-1} \otimes A_2^{-1} \otimes A_3^{-1}$) is still relatively close to the identity, the ideal preconditioned system for this nonsingular Q.

$$
MQ(1:7, 1:7) = \begin{pmatrix} 1.0920 & -0.0193 & -0.0947 & -0.0468 & -0.0727 & -0.0520 & 0.0201 \\ -0.0583 & 1.0465 & -0.0078 & -0.0976 & -0.0173 & -0.0828 & -0.0191 \\ 0.0284 & 0.0121 & 1.1466 & -0.0201 & 0.0649 & -0.0247 & 0.0163 \\ 0.0010 & 0.0283 & -0.0733 & 1.0891 & -0.0346 & 0.0389 & 0.0109 \\ -0.0331 & -0.0685 & 0.0113 & -0.0052 & 1.0676 & -0.0103 & -0.0337 \\ -0.0394 & -0.0597 & -0.0066 & 0.0064 & -0.0455 & 1.0317 & 0.0060 \\ -0.0171 & -0.0010 & -0.0126 & 0.0382 & -0.0166 & 0.0460 & 0.9939 \end{pmatrix}
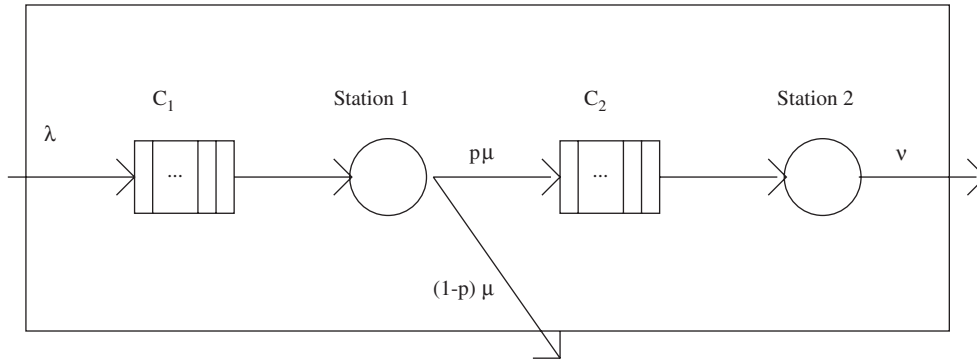$$

Figure 4. SAN queueing network.

Since rough estimates of an approximate inverse often produce effective preconditioners [26], we apply this NKP preconditioner to the well-conditioned system $Qx = e$ to observe its behaviour. The right-hand side was arbitrarily chosen as e. Unpreconditioned full GMRES with a termination criterion that the maximum norm of the residual vector be less than 10^{-8} requires 24 iterations to converge to the solution x, while the NKP preconditioned full GMRES only takes eight iterations. The *ILU* preconditioner with a threshold of .01 converges in five iterations. In terms of the number of iterations, the NKP preconditioner compares favourably with other preconditioners even extending beyond the two-dimensional NKP $A_1 \otimes A_2$ to the three-dimensional NKP $A_1 \otimes A_2 \otimes A_3$.

5.2.2. Small example B. We apply the NKP preconditioner to a small SAN. Thus, the coefficient matrix is singular with rank $n - 1$. We apply the various preconditioners of Section 3 to the SAN described in detail in Reference [12]. This example is a small queueing network consisting of two exponential, finite-capacity, single server stations. See Figure 4.

Two stochastic automata can be used to model this system, which has one synchronizing event and no functional transition rates. We choose the following parameters:

$$\lambda = 13, \quad \mu = 15, \quad v = 11, \quad p = 0.7, \quad C_1 = 9, \quad C_2 = 7$$

This results in a model with 80 states and $N = 2$, $E = 1$.

Below we show the upper left blocks of the three matrices of interest: the SAN descriptor Q, the nearest Kronecker product $A \otimes B$ and the NKP preconditioned iteration matrix MQ, where $M = A^{-1} \otimes B^{-1}$.

$$Q(1:7, 1:7) = \begin{pmatrix} -13 & 0 & 0 & 0 & 0 & 0 & 0 \\ 11 & -24 & 0 & 0 & 0 & 0 & 0 \\ 0 & 11 & -24 & 0 & 0 & 0 & 0 \\ 0 & 0 & 11 & -24 & 0 & 0 & 0 \\ 0 & 0 & 0 & 11 & -24 & 0 & 0 \\ 0 & 0 & 0 & 0 & 11 & -24 & 0 \\ 0 & 0 & 0 & 0 & 0 & 11 & -24 \end{pmatrix}$$

Table I. Number of iterations and CPU times for SAN analysis of queueing network example

Preconditioner	Power method		GMRES method		BiCGSTAB	
	Iterations	Time	Iterations	Time	Iterations	Time
None	1020	0.26	18	1.65	48	0.06
Neumann	343	0.24	11	1.08	21	0.09
Indiv. Inv.	683	0.43	9	0.94	26	0.05
Diagonal	811	0.22	16	1.65	46	0.06
NKP	394	0.16	8	0.87	25	0.04
*ILU*0	177	0.20	9	0.91	15	0.04
ILUTH	57	0.08	6	0.70	3	0.03

$$(A \otimes B)(1:7, 1:7) = \begin{pmatrix} -18.5676 & -0.7431 & 0 & 0 & 0 & 0 & 0 \\ 6.6589 & -25.2265 & -0.7431 & 0 & 0 & 0 & 0 \\ 0 & 6.6589 & -25.2265 & -0.7431 & 0 & 0 & 0 \\ 0 & 0 & 6.6589 & -25.2265 & -0.7431 & 0 & 0 \\ 0 & 0 & 0 & 6.6589 & -25.2265 & -0.7431 & 0 \\ 0 & 0 & 0 & 0 & 6.6589 & -25.2265 & -0.7431 \\ 0 & 0 & 0 & 0 & 0 & 6.6589 & -25.2265 \end{pmatrix}$$

$$MQ(1:7, 1:7) = \begin{pmatrix} 0.6708 & -0.2429 & 0.0071 & -0.0002 & 0 & 0 & 0 \\ -0.2866 & 0.8937 & -0.1774 & 0.0052 & -0.0002 & 0 & 0 \\ -0.0751 & -0.2282 & 0.9109 & -0.1779 & 0.0052 & -0.0002 & 0 \\ -0.0197 & -0.0598 & -0.2237 & 0.9108 & -0.1779 & 0.0052 & -0.0002 \\ -0.0052 & -0.0157 & -0.0586 & -0.2238 & 0.9108 & -0.1779 & 0.0052 \\ -0.0013 & -0.0041 & -0.0154 & -0.0586 & -0.2238 & 0.9108 & -0.1780 \\ -0.0004 & -0.0011 & -0.0040 & -0.0153 & -0.0585 & -0.2232 & 0.9128 \end{pmatrix}$$

The NKP, $A \otimes B$, captures the structure of Q rather well and the elements in the NKP are close enough to those in Q to give an iteration matrix MQ with a diagonally dominant banded structure. While one might hope MQ is closer to $I - e\pi$, the ideal preconditioned system for a singular, rank $n-1$ system, the NKP preconditioner still serves its purpose of improving the convergence of the iterative methods. Table I clearly shows the success of the two-dimensional NKP preconditioner on this SAN. (An NKP preconditioner with N matrices $A_1 \otimes A_2 \otimes \cdots \otimes A_N$ is called an N-dimensional NKP.) All experiments were performed on an SUN Ultra 10 workstation running MATLAB programs. The iterative methods used were the power method, iterative GMRES with restarts and BiCGSTAB. For GMRES, the size of the subspace was fixed at 10. For the Neumann preconditioner, we set $H = 2$. All iterative procedures stop as soon as the maximum norm of the residual vector becomes smaller than 10^{-8}.

The NKP preconditioner beats the other SAN preconditioners: the Neumann, the individual inverse and the diagonal preconditioners. The NKP preconditioner wins in terms of time and, with the exception of the Neumann preconditioner, it always converges in fewer iterations. The Neumann preconditioner is known to provide a theoretically good approximation to $Q^{\#}$; however, it is also known to require a great deal of computation time. In contrast, the NKP preconditioner is relatively cheap to compute, as is quantified in a subsequent paper [23], and often reduces the number of iterations drastically. Note that the NKP preconditioner does quite well in comparison to the *ILU* preconditioners, which are generally known to be the best preconditioners for Markov chains. However, the problem with using *ILU* preconditioners for SANs is that the approximate *L*, *U* factors are hard to obtain due to the structure of SANs. (Here, for comparison purposes, we simply expanded the SAN descriptor into its two-dimensional matrix form and found its approximate *L*, *U* factors. This was possible due to the small order of *Q*.) The NKP preconditioner has no such structural problems; it incorporates the SAN structure naturally, making it, as is further demonstrated in a subsequent paper, the leader in SAN preconditioning.

6. CONCLUSION

We have discussed the method for finding the approximate NKP for any matrix of the form $Q = \sum_{j=1}^{T} \otimes_{i=1}^{N} Q_j^{(i)}$. We used this approximate NKP as the basis for a preconditioner. Testing on small artificial examples, we found several instances where the NKP preconditioner worked well. While extensive testing on larger matrices, arising most especially from Markov chains and SANs remains to be done, we have provided the essential theoretical framework for such practical exploration. Future research might reveal why the NKP works well on some matrices and not others. Are there specific properties of the input matrices, $Q_j^{(i)}$'s, which will determine *a priori* the success of the NKP as a preconditioner? Is there a practical limit on N, the number of small matrices in the Kronecker product, for which the approximation of the NKP becomes too gross? How does the singularity of the infinitesimal generator matrix Q of a Markov chain affect the formation of the NKP? And finally, it would be interesting to compare by means of a small three-dimensional tensor the approximate NKP induced by truncating the HOSVD with the exact NKP derived from de Lathauwer's best rank-1 algorithm.

REFERENCES

1. Plateau B. On the stochastic structure of parallelism and synchronization models for distributed algorithms. *Performance Evaluation Review* 1985; **13**:142–154.
2. Plateau B, Atif K. Stochastic automata network for modelling parallel systems. *IEEE Transactions on Software Engineering* 1991; **17**(10):1093–1108.
3. Plateau B, Fourneau JM. A methodology for solving Markov models of parallel systems. *Journal of Parallel and Distributed Computing* 1991; **12**:370–387.
4. Stewart WJ. *Introduction to the Numerical Solution of Markov Chains*. Princeton University Press: Princeton, 1994.
5. Atif K. Modelisation du parallelisme et de la synchronisation. *Ph.D. Thesis*, l'Institut National Polytechnique de Grenoble, Grenoble, France, 1988.
6. Buchholz P. Equivalence relations for stochastic automata networks. In *Computations with Markov Chains*, Stewart WJ (ed.). Kluwer Academic: Boston, 1995, 197–215.
7. Philippe B, Saad Y, Stewart WJ. Numerical methods in Markov chain modeling. *Operations Research* 1992; **40**(6):1156–1179.
8. Stewart WJ, Wu W. Numerical experiments with iteration and aggregation for Markov chains. *ORSA Journal on Computing* 1992; **4**(3):336–350.

9. Dayar T. State space orderings for Gauss–Seidel in Markov chains revisited. *SIAM Journal of Scientific Computing* 1998; **19**(1):148–154.
10. Uysal E, Dayar T. Iterative methods based on splittings for stochastic automata networks. *European Journal of Operational Research* 1998; **110**:166–186.
11. Fernandes P, Plateau B, Stewart WJ. Efficient descriptor-vector multiplications in stochastic automata networks. *Journal of Association for Computing Machinery* 1998; **45**(3):381–414.
12. Stewart WJ, Atif K, Plateau B. The numerical solution of stochastic automata networks. *European Journal of Operational Research* 1995; **86**:503–525.
13. Buchholz P. Projection methods for the analysis of stochastic automata networks. In *Numerical Solution of Markov Chains*, Plateau B, Stewart WJ, Silva M (eds). Prensas Universitarias de Zaragoza: Zaragoza, 1999, 149–168.
14. Saad Y. Preconditioned Krylov subspace methods for the numerical solution of Markov chains. In *Computations with Markov Chains*, Stewart WJ (ed.). Kluwer Academic: Boston, 49–64.
15. Plateau B. PEPS: A package for solving complex Markov models of parallel systems. In *Modelling Techniques and Tools for Computer Performance Evaluation*, Puigjaner R, Potier D (eds). Plenum Press: New York, 1990; 291–306.
16. Benzi M, Tuma M. A Parallel Solver for Large-Scale Markov Chains. *Applied Numerical Mathematics* 2002; **41**:135–153.
17. Meyer CD. The role of the group generalized inverse in the theory of finite Markov chains. *SIAM Review* 1975; **17**(3):443–464.
18. Plateau B, Stewart WJ, Fernandes P. On the benefits of using functional transitions in Kronecker modelling. *Performance Evaluation* 2001; submitted.
19. A. Greenbaum. *Iterative Methods for Solving Linear Systems*. SIAM: Philadelphia, 1997.
20. Benzi M, Meyer CD, Tuma M. A sparse approximate inverse preconditioner for the conjugate gradient method. *SIAM Journal on Scientific Computing* 1996; **17**:1135–1149.
21. Kolotilina LY, Yeremin AY. Factorized sparse approximate inverse preconditioning I: theory. *SIAM Journal of Matrix Analysis and its Applications* 1993; **14**:45–58.
22. Pitsianis N, Van Loan CF. Approximation with Kronecker products. In *Linear Algebra for Large Scale and Real Time Applications*, Moonen MS, Golub GH (eds). Kluwer Academics: Boston, 1993; 293–314.
23. Langville AN, Stewart WJ. Testing the nearest Kronecker product preconditioner on Markov chains and stochastic automata networks. INFORMS *Journal on Computing*. Accepted May 2003, forthcoming.
24. Huyer W, Neumaier A. Global optimization by multilevel coordinate search. *Journal of Global Optimization* 1999; **14**:331–355.
25. Barrett R, Berry M, Chan TF *et al. Templates for the Solution of Linear Systems*: *Building Blocks for Iterative Methods*. SIAM: Philadelphia, 1994.
26. Philippe B, Sidje RB. Transient solutions of Markov processes by Krylov subspaces. In *Computations with Markov Chains*, Stewart WJ (ed.). Kluwer Academic: Boston, 1995; 95–120.
27. Campbell SL, Meyer CD. *Generalized Inverses of Linear Transformations*. Pitman: London, 1979.
28. de Lathauwer L. Signal processing based on multilinear algebra. *Ph.D. Thesis*, Faculty of Engineering, K. U. Leuven (Leuven, Belgium), 1997.
29. de Moor B, de Lathauwer L, Vandewalle J. A multilinear singular value decomposition. *SIAM Journal of Matrix Analysis and Applications* 2000; **21**(4):1253–1278.
30. de Moor B, de Lathauwer L, Vandewalle J. On the best rank-1 and rank-$(R_1, R_2, \ldots, R_N)$ approximation of higher-order tensors. *SIAM Journal of Matrix Analysis and Applications* 2000; **21**(4):1324–1342.
31. Van Loan CF. The ubiquitous Kronecker product. *Journal of Computational and Applied Mathematics* 2000; **123**:85–100.