SCIENCE CHINA
Mathematics

• ARTICLES •

June 2012 Vol. 55 No. 6: 1281–1292
doi: 10.1007/s11425-012-4363-5

A projection method and Kronecker product preconditioner for solving Sylvester tensor equations

CHEN Zhen[1,2] & LU LinZhang[2,1,*]

[1]*School of Mathematical Sciences, Xiamen University, Xiamen* 361005*, China;*
[2]*School of Mathematics and Computer Science, Guizhou Normal University, Guiyang* 550001*, China*
Email: zchen@gznu.edu.cn, lzlu@xmu.edu.cn

Received March 4, 2011; accepted December 7, 2011; published online February 17, 2012

Abstract The preconditioned iterative solvers for solving Sylvester tensor equations are considered in this paper. By fully exploiting the structure of the tensor equation, we propose a projection method based on the tensor format, which needs less flops and storage than the standard projection method. The structure of the coefficient matrices of the tensor equation is used to design the nearest Kronecker product (NKP) preconditioner, which is easy to construct and is able to accelerate the convergence of the iterative solver. Numerical experiments are presented to show good performance of the approaches.

Keywords Sylvester tensor equation, Schur decomposition, projection method, nearest Kronecker product (NKP), preconditioning

MSC(2010) 65F10, 15A69

Citation: Chen Z, Lu L Z. A projection method and Kronecker product preconditioner for solving Sylvester tensor equations. Sci China Math, 2012, 55(6): 1281–1292, doi: 10.1007/s11425-012-4363-5

1 Introduction

In this paper, we consider the solution of Sylvester tensor equation of form

$$\mathcal{X} \times_1 A^{(1)} + \mathcal{X} \times_2 A^{(2)} + \cdots + \mathcal{X} \times_N A^{(N)} = \mathcal{D}, \tag{1}$$

where the known matrices $A^{(n)} \in \mathbb{R}^{I_n \times I_n} (n = 1, 2, \ldots, N)$, tensor $\mathcal{D} \in \mathbb{R}^{I_1 \times I_2 \times \cdots \times I_N}$, and the unknown tensor $\mathcal{X} \in \mathbb{R}^{I_1 \times I_2 \times \cdots \times I_N}$, which is called N-mode tensor. The properties of the operators $\times_1$, $\times_2$, …, $\times_N$ will be described in detail in Section 2. Throughout the paper, we refer to Equation (1) as a Sylvester tensor equation, since when $\mathcal{X}$ is a simple 2-mode tensor, i.e., a matrix, (1) can be reduced to the Sylvester matrix equation

$$AX + XB^{\mathrm{T}} = D, \tag{2}$$

which arises frequently from the areas of systems and control theory [6–8] and has received much attention (see [12, 21, 26]). In the case that $\mathcal{X}$ is a 3-mode tensor, (1) becomes

$$\mathcal{X} \times_1 A + \mathcal{X} \times_2 B + \mathcal{X} \times_3 C = \mathcal{D}, \tag{3}$$

*Corresponding author

which usually comes from the finite element [10], finite difference [3] or spectral method [19, 20] discretization of a linear partial differential equation in high dimension. For example, by the discretization of three-dimensional radiative transfer equation using the Chebyshev collocation spectral method, the resultant algebraic system is of form (3) (see [20]).

Recently, Li et al. [20] reformulated (3) as a sequence of Sylvester matrix equations (2) by applying Schur decomposition and then solved the resulting linear system by a back-substitution process. This approach is efficient for small sized coefficient matrices and can compute the exact solution after a finite number of steps in the absence of roundoff error. However, we note that it is difficult to generalize the approach to the case of $N > 3$ (see an illustration in Section 3). For large and sparse coefficient matrices, such kinds of direct methods are often too expensive to implement, and iterative methods should be more practical choices. In [5], Beylkin and Mohlenkamp reformulated (1) as a linear squares problem. For a given $\mathcal{X}$, they solved the associated normal equations by fixing all but one direction. This procedure is known as an alternating least squares (ALS) approach which converges rather slowly in most cases. Kressner and Tobler [15] proposed a so-called tensor Krylov subspace method for solving (1). They constructed a tensorised Krylov space which is the Kronecker product of the usual Krylov subspaces, then transformed the original linear system to a system of smaller size. Ballani and Grasedyck [4] presented an iterative scheme similar to Krylov subspace method relying on truncation operator, whereas the operator is implemented by hierarchical Tucker format [9]. These methods are constructive, however, the approaches are only applicable for certain special coefficient matrices and the right-hand side.

In contrast with its equivalent linear system, Sylvester tensor equation (1) has more compact structure which should be made fully use of. In this paper, we propose a projection method based on tensor format (PM_BTF) for solving (1). The proposed method needs not to form the coefficient matrix explicitly and only relies on tensor-matrix multiplication, which is more effective than the matrix-vector multiplication in solving process. Hence less flops and storage than the standard projection method (SPM) are needed. Furthermore, we develop the nearest Kronecker product (NKP) preconditioner to accelerate convergence of the algorithms. The NKP preconditioner can be constructed easily and work effectively. It is based on the result that the summation of several Kronecter products can be approximated by one Kronecker product [16]. Similar preconditioners can be found in [17, 22, 28].

The rest of this paper is organized as follows. Section 2 contains some tensor notations and basic operations used throughout the paper. In Section 3, we recall the Schur decomposition method for solving (3). A projection method based on tensor format for solving (1) is proposed in Section 4. In Section 5, we design appropriate NKP preconditioner for (1). The numerical experiments in Section 6 demonstrate the performance of the algorithms. Finally, conclusions are given in Section 7.

Throughout the paper, we use the following notations. Matrices are denoted by capital letters, e.g., A, B, C. Tensors are denoted by Euler script letters, e.g., $\mathcal{X}$, $\mathcal{Y}$, $\mathcal{Z}$. The $n \times n$ identity matrix is denoted by $I^{(n)}$. The order of a tensor is the number of dimensions, also known as ways or modes. As a special case, vector is 1-mode tensor and matrix is 2-mode tensor. The operator $\mathrm{vec}(\cdot)$ stacks the columns of a matrix (or a tensor) to form a vector, $\mathrm{tr}(\cdot)$ denotes the trace of matrix. $\otimes$ denotes the Kronecker product and $\|\cdot\|$ denotes the Frobenius norm.

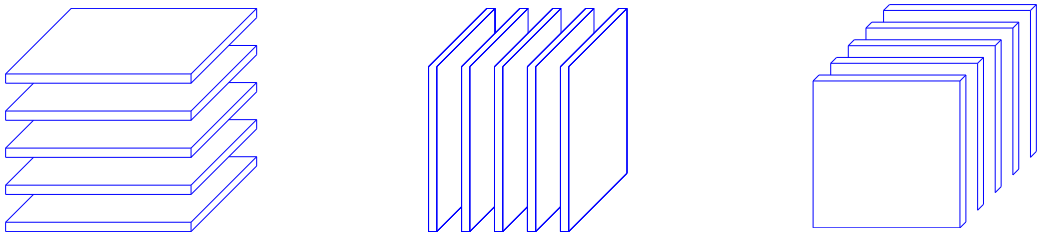
2 Tensor notations and basic operations

In this section we briefly review some tensor arithmetic concepts and notations that are needed in the rest of the paper. For more details on properties of tensor, see recent reviews [14, 18, 24].

2.1 Unfolding and n-mode product

The unfolding of a tensor $\mathcal{X}$ along mode n is an $I_n \times (I_{n+1} \cdots I_N I_1 \cdots I_{n-1})$ matrix denoted by $X_{(n)}$, whose column is a column of $\mathcal{X}$ along the n-th mode. A tensor slice is the two-dimensional section of a tensor, obtained by fixing all but two indices [14]. Figure 1 shows the horizontal, lateral, and frontal

slices of a 3-mode tensor $\mathcal{X}$, denoted by $X_{i::}$, $X_{:j:}$, and $X_{::k}$, respectively. The k-th frontal slice of a 3-mode tensor $X_{::k}$ can also be denoted by X_k compactly.



(a) Horizontal slices: $X_{i::}$ (b) Lateral slices: $X_{:j:}$ (c) Frontal slices: $X_{::k}$ (or X_k)

Figure 1 Slices of a 3-mode tensor $\mathcal{X}$

An important kind of operation for a tensor is the tensor-matrix multiplication, also known as n-mode (matrix) product. The n-mode product of a tensor $\mathcal{X} \in \mathbb{R}^{I_1 \times I_2 \times \cdots \times I_N}$ with a matrix $A \in \mathbb{R}^{J \times I_n}$ is denoted by $\mathcal{X} \times_n A$. The result is of size $I_1 \times \cdots \times I_{n-1} \times J \times I_{n+1} \times \cdots \times I_N$ and

$$(\mathcal{X} \times_n A)_{i_1 \cdots i_{n-1} j i_{n+1} \cdots i_N} = \sum_{i_n=1}^{I_n} x_{i_1 i_2 \cdots i_N} a_{j i_n}.$$

The n-mode (vector) product of a tensor $\mathcal{X} \in \mathbb{R}^{I_1 \times I_2 \times \cdots \times I_N}$ with a vector $v \in \mathbb{R}^{I_n}$ is denoted by $\mathcal{X} \bar{\times}_n v$. The result is an $(N-1)$-mode tensor and

$$(\mathcal{X} \bar{\times}_n v)_{i_1 \cdots i_{n-1} i_{n+1} \cdots i_N} = \sum_{i_n=1}^{I_n} x_{i_1 i_2 \cdots i_N} v_{i_n}.$$

In terms of the definition of unfolding, we obtain

$$\begin{aligned} \mathcal{Y} &= \mathcal{X} \times_1 A^{(1)} \times_2 A^{(2)} \cdots \times_N A^{(N)} \\ &\Leftrightarrow Y_{(n)} = A^{(n)} X_{(n)} (A^{(N)} \otimes \cdots \otimes A^{(n+1)} \otimes A^{(n-1)} \otimes \cdots \otimes A^{(1)})^{\mathrm{T}}. \end{aligned}$$

It follows immediately from the definition that n-mode and m-mode multiplication commute if $m \neq n$:

$$\mathcal{X} \times_m A \times_n B = \mathcal{X} \times_n B \times_m A.$$

If the modes are the same, we have

$$\mathcal{X} \times_n A \times_n B = \mathcal{X} \times_n (BA).$$

2.2 Inner product and tensor norm

The inner product of the two tensors $\mathcal{X}, \mathcal{Y} \in \mathbb{R}^{I_1 \times I_2 \times \cdots \times I_N}$ is defined by

$$\langle \mathcal{X}, \mathcal{Y} \rangle = \mathrm{vec}(\mathcal{X})^{\mathrm{T}} \mathrm{vec}(\mathcal{Y}) = \sum_{i_1=1}^{I_1} \sum_{i_2=1}^{I_2} \cdots \sum_{i_N=1}^{I_N} x_{i_1 i_2 \cdots i_N} y_{i_1 i_2 \cdots i_N},$$

and the norm induced by this inner product is

$$\|\mathcal{X}\| = \sqrt{\langle \mathcal{X}, \mathcal{X} \rangle}.$$

For the two tensors $\mathcal{X}$ and $\mathcal{Y}$, the computation of $\|\mathcal{X} - \mathcal{Y}\|$ can be simplified as follows:

$$\|\mathcal{X} - \mathcal{Y}\|^2 = \|\mathcal{X}\|^2 - 2\langle \mathcal{X}, \mathcal{Y} \rangle + \|\mathcal{Y}\|^2.$$

Moreover, n-mode multiplication commutes with respect to the inner product, i.e.,

$$\langle \mathcal{X}, \mathcal{Y} \times_n A \rangle = \langle \mathcal{X} \times_n A^{\mathrm{T}}, \mathcal{Y} \rangle.$$

3 Schur decomposition for solving (3)

In this section, we briefly review the direct method based on Schur decomposition [20] for solving (3). Note that here we make full use of tensor arithmetic concepts and notations which are different from [20].

Let $\mathcal{X} \in \mathbb{R}^{m\times n\times l}$. The matrices A, B and C can be reduced to lower (quasi-lower) triangular forms by using real Schur decomposition

$$A' = U^{\mathrm{T}}AU, \quad B' = V^{\mathrm{T}}BV, \quad C' = W^{\mathrm{T}}CW,$$

where U, V and W are orthogonal matrices. Multiplying Equation (3) along 1, 2 and 3-mode by U^{T}, V^{T} and W^{T} yields

$$\mathcal{X}' \times_1 A' + \mathcal{X}' \times_2 B' + \mathcal{X}' \times_3 C' = \mathcal{D}', \tag{4}$$

where $\mathcal{X}' = \mathcal{X} \times_1 U^{\mathrm{T}} \times_2 V^{\mathrm{T}} \times_3 W^{\mathrm{T}}$ and $\mathcal{D}' = \mathcal{D} \times_1 U^{\mathrm{T}} \times_2 V^{\mathrm{T}} \times_3 W^{\mathrm{T}}$. For simplicity, we omit the superscript of (4) in the rest of this section.

Unfolding Equation (4) along the mode 1, we obtain

$$AX_{(1)} + I^{(m)}X_{(1)}(I^{(l)} \otimes B)^{\mathrm{T}} + I^{(m)}X_{(1)}(C \otimes I^{(n)})^{\mathrm{T}} = D_{(1)}. \tag{5}$$

In the blocked form, the above equation can be transformed into a sequence of l matrix equations as follows

$$AX_k + X_kB^{\mathrm{T}} + \sum_{t=1}^{l} c_{kt}X_t = D_k, \quad k = 1, 2, \ldots, l, \tag{6}$$

where X_k is the k-th frontal slice of tensor $\mathcal{X}$.

In order to solve (6), having in mind that C is quasi-lower triangular matrix, the following two cases are considered.

Case 1. The k-th eigenvalue of C is real. This implies that $c_{k,k+1} = 0$. Set $F_k = D_k - \sum_{t=1}^{k-1} c_{kt}X_t$, then (6) can be reduced to the compact form

$$AX_k + X_k(B^{\mathrm{T}} + c_{kk}I^{(n)}) = F_k, \tag{7}$$

which amounts to a Sylvester equation and can be solved by standard techniques [8].

Case 2. The k-th eigenvalue of C is complex. This implies that $c_{k,k+1} \neq 0$. In this case, (6) can be reduced to the following form

$$\begin{cases} AX_k + X_k(B^{\mathrm{T}} + c_{kk}I^{(n)}) + c_{k,k+1}X_{k+1} = F_k, \\ c_{k+1,k}X_k + AX_{k+1} + X_{k+1}(B^{\mathrm{T}} + c_{k+1,k+1}I^{(n)}) = F_{k+1}, \end{cases} \tag{8}$$

which also amounts to solving a Sylvester equation on the two unknown matrices X_k and X_{k+1} (more algorithm details can be found in [20]).

The above process is summarized in Algorithm 3.1.

Algorithm 3.1 The direct method based on Schur decomposition (DM_BSD)

Input: Coefficients A, B, C, and $\mathcal{D}$ of Equation (3)

Output: The solution $\mathcal{X}$ of (3)

Compute real Schur decomposition of A, B, and C.

Set $A := U^{\mathrm{T}}AU$, $B := V^{\mathrm{T}}BV$, $C := W^{\mathrm{T}}CW$, $\mathcal{D} := \mathcal{D} \times_1 U^{\mathrm{T}} \times_2 V^{\mathrm{T}} \times_3 W^{\mathrm{T}}$.

For $k = 1 : l$ do

If $c_{k,k+1} = 0$ then

Apply (7) to compute X_k.

Else if $c_{k,k+1} \neq 0$ then

Apply (8) to compute X_k, X_{k+1}.

$k \leftarrow k+2$

End if

End for

Construct $X_{(1)} := [X_1, X_2, \ldots, X_l]$, $\mathcal{X} := \text{folding}(X_{(1)})$.

$\mathcal{X} := \mathcal{X} \times_1 U \times_2 V \times_3 W$.

However, Algorithm 3.1 is difficult to generalize to the case of $N > 3$. To show this, let us assume that $N = 4$ and $I_n = 2$, in this case, (1) has the form

$$AX_{(1)} + IX_{(1)}(I \otimes\ I \otimes B)^{\mathrm{T}} + IX_{(1)}(I \otimes C \otimes I)^{\mathrm{T}} + IX_{(1)}(D \otimes I \otimes I)^{\mathrm{T}} = E_{(1)},$$

where I is the 2×2 identity matrix. By applying the definition of unfolding and MATLAB notation, the above summands have the following form

$$\begin{aligned}
&AX_{(1)} = [AX_{::11},\ AX_{::21},\ AX_{::12},\ AX_{::22}],\\
&IX_{(1)}(I \otimes\ I \otimes B)^{\mathrm{T}} = \left[X_{::11}B^{\mathrm{T}},\ X_{::21}B^{\mathrm{T}},\ X_{::12}B^{\mathrm{T}},\ X_{::22}B^{\mathrm{T}}\right],\\
&IX_{(1)}(I \otimes C \otimes I)^{\mathrm{T}} = [X_{::11},\ X_{::21},\ X_{::12},\ X_{::22}] \begin{bmatrix} C^{\mathrm{T}} & \mathbf{0} \\ \mathbf{0} & C^{\mathrm{T}} \end{bmatrix} \otimes I,\\
&IX_{(1)}(D \otimes I \otimes I)^{\mathrm{T}} = [X_{::11},\ X_{::21},\ X_{::12},\ X_{::22}] \begin{bmatrix} d_{11}I & d_{21}I \\ d_{12}I & d_{22}I \end{bmatrix} \otimes I.
\end{aligned}$$

It is obvious that the above equations have not Sylvester's structure similar to (6) or (8). Therefore, it is difficult to extend the Schur decomposition method to the case of $N > 3$.

4 A projection method in tensor format

In this section, we will develop an iterative algorithm for approximating the solution $\mathcal{X}$ to the Sylvester tensor equation (1). Firstly, we consider the existence and uniqueness of the solution of (1). By using the Kronecker product [4, 15], (1) can be reformulated as follows:

$$Ax = b, \tag{9}$$

with

$$A = I^{(I_N)} \otimes \cdots \otimes I^{(I_2)} \otimes A^{(1)} + \cdots + A^{(N)} \otimes I^{(I_{N-1})} \otimes \cdots \otimes I^{(I_1)}, \tag{10}$$

and $x = \text{vec}(\mathcal{X})$, $b = \text{vec}(\mathcal{D})$. The following lemma is a well-known property of the Kronecker product [9].

Lemma 4.1. *Consider the matrix A defined in* (10)*, then the set of all the eigenvalues of A, denoted by $\sigma(A)$, is given by all possible sums of eigenvalues of $A^{(1)}, A^{(2)}, \ldots, A^{(N)}$:*

$$\sigma(A) = \{\lambda_1 + \lambda_2 + \cdots + \lambda_N |\, \lambda_n \in \sigma(A^{(n)})\}.$$

By Lemma 4.1, it immediately leads to the following result.

Lemma 4.2. *Equation* (1) *has a unique solution if and only if*

$$\lambda_1 + \lambda_2 + \cdots + \lambda_N \neq 0, \quad \forall\, \lambda_n \in \sigma(A^{(n)}).$$

For the Sylvester matrix equation (2), the result is corresponding to the well-known condition $\sigma(A) \cap \sigma(-B) = \emptyset$.

Now, apply the well-known generalized minimal residuals (GMRES) method to solving (9), we have the following algorithm.

Algorithm 4.1 GMRES (see [25])

Input: Coefficients A, and b of the linear system (9)

Output: Approximate solution x_m of (9)

1. Compute $r_0 = b - Ax_0$, $\beta := \|r_0\|_2$, and $v_1 := r_0/\beta$.
2. Define the $(m+1) \times m$ matrix $H_m = (h_{ij})$. Set $H_m = 0$.
3. For $j = 1, 2, \ldots, m$ do
4. Compute $w_j := Av_j$
5. For $i = 1, 2, \ldots, j$ do
6. $h_{ij} := \langle w_j, v_i \rangle$
7. $w_j := w_j - h_{ij} v_i$
8. End for
9. $h_{j+1,j} = \|w_j\|_2$. If $h_{j+1,j} = 0$ set $m := j$ and go to 12.
10. $v_{j+1} = w_j / h_{j+1,j}$
11. End for
12. Compute y_m the minimizer of $\|\beta e_1 - H_m y\|_2$ and $x_m = x_0 + V_m y_m$.

In Algorithm 4.1, we assume that the dimension of the space $\mathcal{V}$ is a fixed number m. In the practical GMRES method, this parameter is usually chosen dynamically [25]. In addition, it is important to note that $A \in \mathbb{R}^{I^N \times I^N}, x \in \mathbb{R}^{I^N}$ and the computational cost of each step of matrix-vector product in line 4 is $\mathcal{O}(I^{2N})$, which is quite large. If we compute the tensor-matrix multiplication $\mathcal{V}_j \times_1 A^{(1)} + \cdots + \mathcal{V}_j \times_N A^{(N)}$ instead of matrix-vector product Av_j, where $\mathcal{V}_j$ is an N-mode tensor reshaped by v_j, then the complexity $\mathcal{O}(I^{2N})$ can be reduced to $\mathcal{O}(NI^{N+1})$, which is much smaller, especially for large N. Furthermore, Algorithm 4.1 needs to form the coefficient matrix (10) explicitly, which is impracticable. These facts drive us to derive the tensor format GMRES by applying tensor format.

Algorithm 4.2 GMRES based on tensor format (GMRES_BTF)

Input: Coefficients $A^{(1)}, A^{(2)}, \ldots, A^{(N)}$, and $\mathcal{D}$ of Equation (1)

Output: Approximate solution $\mathcal{X}_m$ of (1)

1. Compute $\mathcal{R}_0 = \mathcal{D} - (\mathcal{X}_0 \times_1 A^{(1)} + \cdots + \mathcal{X}_0 \times_N A^{(N)})$, $\beta := \|\mathcal{R}_0\|$, and $\mathcal{V}_1 := \mathcal{R}_0/\beta$.
2. Define the $(m+1) \times m$ matrix $H_m = (h_{ij})$. Set $H_m = 0$.
3. For $j = 1, 2, \ldots, m$ do
4. Compute $\mathcal{W}_j := \mathcal{V}_j \times_1 A^{(1)} + \cdots + \mathcal{V}_j \times_N A^{(N)}$
5. For $i = 1, 2, \ldots, j$ do
6. $h_{ij} := \langle \mathcal{W}_j, \mathcal{V}_i \rangle$
7. $\mathcal{W}_j := \mathcal{W}_j - h_{ij} \mathcal{V}_i$
8. End for
9. $h_{j+1,j} = \|\mathcal{W}_j\|$. If $h_{j+1,j} = 0$ set $m := j$ and go to 12.
10. $\mathcal{V}_{j+1} = \mathcal{W}_j / h_{j+1,j}$
11. End for

12. Compute y_m the minimizer of $\|\beta e_1 - H_m y\|_2$ and $\mathcal{X}_m = \mathcal{X}_0 + \widetilde{\mathcal{V}} \bar{\times}_{N+1} y_m$, where $\widetilde{\mathcal{V}}$ is an $(N+1)$-mode tensor with column tensors $\mathcal{V}_1, \mathcal{V}_2, \ldots, \mathcal{V}_m$.

The algorithm based on the tensor format for solving (1) is not only suitable for GMRES method, but also applicable to almost all projection methods such as biconjugate gradients (BiCG), stabilized BiCG (BiCGSTAB), conjugate gradients squared (CGS), quasi-minimal residuals (QMR), transpose-free QMR (TFQMR). The advantage of application of tensor format is confirmed by the numerical experiments in Section 6.

5 Preconditioning

It is well known that an iterative algorithm for the linear system can perform better when a good preconditioner is used. Specifically, suppose we wish to solve the nonsingular system (9). For any nonsingular matrix M, the system

$$MAx = Mb \tag{11}$$

has the same solution with (9). The convergence of the iteration method for solving (11) will depend on the properties of MA instead of those of A. In particular, if M is a good approximation of A^{-1}, (11) can be solved more effectively and efficiently than (9).

In (10), A is given as the sum of N Kronecker products of matrices. The Kronecker product has some important properties [9]. For example, $\mathrm{tr}(F \otimes G) = \mathrm{tr}(F) \otimes \mathrm{tr}(G)$, $(F \otimes G)^{-1} = F^{-1} \otimes G^{-1}$. When $N = 2$, Pitsianis and Van Loan [23] presented a method for finding the nearest Kronecker production (NKP), $Q \otimes P$, for A. Since $Q \otimes P \approx A$, one would hope that $(Q \otimes P)^{-1} \approx A^{-1}$ and thus define $M = Q^{-1} \otimes P^{-1}$ as the preconditioner. Recall that the NKP problem for $N > 2$ can be written as follows: find Q_1, Q_2, …, Q_N such that

$$\|A - Q_1 \otimes Q_2 \otimes \cdots \otimes Q_N\|^2 = \left\| \sum_{j=1}^{N} \bigotimes_{i=1}^{N} A_j^{(i)} - Q_1 \otimes Q_2 \otimes \cdots \otimes Q_N \right\|^2$$

is minimized. In [16], Langville and Stewart proved that

$$\begin{cases} Q_1 \approx \alpha_{11} A_1^{(1)} + \alpha_{12} A_2^{(1)} + \cdots + \alpha_{1N} A_N^{(1)}, \\ Q_2 \approx \alpha_{21} A_1^{(2)} + \alpha_{22} A_2^{(2)} + \cdots + \alpha_{2N} A_N^{(2)}, \\ \vdots \\ Q_N \approx \alpha_{N1} A_1^{(N)} + \alpha_{N2} A_2^{(N)} + \cdots + \alpha_{NN} A_N^{(N)}. \end{cases}$$

This statement allows us to take $M = Q_1^{-1} \otimes Q_2^{-1} \otimes \cdots \otimes Q_N^{-1}$ as the preconditioner for solving (9) because finding the small Q_1^{-1}, Q_2^{-1}, …, Q_N^{-1} is not difficult and M does not need to form explicitly. The preconditioning effect of the NKP preconditioner is completely determined by the spectral distribution of the matrix MA. It is confirmed by the numerical experiments in Section 6. In comparison with other preconditioners such as ILU, Neumann, diagonal, the NKP preconditioner enjoys considerable efficiency [17].

Now, the remaining question is how to compute the unknown parameters α_{ij} $(i, j = 1, 2, \ldots, N)$. From trace definition of the Frobenius norm, i.e., $\|A\| = \sqrt{\mathrm{tr}(A^{\mathrm{T}} A)}$, and the above statement, we can transform the minimal problem into [16],

$$\left\| \sum_{j=1}^{N} \bigotimes_{i=1}^{N} A_j^{(i)} - Q_1 \otimes Q_2 \otimes \cdots \otimes Q_N \right\|^2$$

$$\begin{aligned}
&\approx \Big\| \sum_{j=1}^{N} \bigotimes_{i=1}^{N} A_j^{(i)} - \Big(\sum_{j=1}^{N} \alpha_{1j} A_j^{(1)} \Big) \otimes \Big(\sum_{j=1}^{N} \alpha_{2j} A_j^{(2)} \Big) \otimes \cdots \otimes \Big(\sum_{j=1}^{N} \alpha_{Nj} A_j^{(N)} \Big) \Big\|^2 \\
&= \Big[\sum_{i,j=1}^{N} \prod_{k=1}^{N} \mathrm{tr}(A_i^{(k)T} A_j^{(k)}) \Big] - 2 \Big[\sum_{i=1}^{N} \Big(\prod_{k=1}^{N} \Big(\sum_{j=1}^{N} \alpha_{kj} \mathrm{tr}(A_i^{(k)T} A_j^{(k)}) \Big) \Big) \Big] \\
&\quad + \prod_{k=1}^{N} \Big[\sum_{i,j=1}^{N} \alpha_{ki} \alpha_{kj} \mathrm{tr}(A_i^{(k)T} A_j^{(k)}) \Big]. \qquad (12)
\end{aligned}$$

(12) means that finding the NKP matrices approximately is equivalent to finding α_{ij} $(i, j = 1, 2, \ldots, N)$ such that the above nonlinear function is minimized. Clearly, as N increases, this minimization problem which includes N^2 variables becomes impractical. Fortunately, combining the above statements with the structure of the coefficient matrix A in (10), we have the following proposition.

Proposition 5.1. *Let A be of the form* (10). *Then the approximate NKP matrices $Q_1, Q_2, \ldots, Q_N$ such that $A \approx Q_1 \otimes Q_2 \otimes \cdots \otimes Q_N$ are as follows*:

$$\begin{cases}
Q_1 \approx \alpha_{11} A^{(N)} + \alpha_{12} I^{(I_N)}, \\
Q_2 \approx \alpha_{21} A^{(N-1)} + \alpha_{22} I^{(I_{N-1})}, \\
\vdots \\
Q_N \approx \alpha_{N1} A^{(1)} + \alpha_{N2} I^{(I_1)}.
\end{cases}$$

Proposition 5.1 shows that the number of unknown parameters reduces from N^2 to $2N$, and thereby, the complexity of the nonlinear optimization problem (12) is reduced. Using the nonlinear optimization software such as `fminsearch` in MATLAB, `nlp` in SAS, multilevel coordinate search (MCS) in [13], the optimal parameters α_{ij} can be found quickly. Therefore, the application of GMRES method to (11) yields the following preconditioned version of GMRES [25].

Algorithm 5.1 GMRES_BTF with the NKP preconditioner (GMRES_BTF+NKP)

Input: Coefficients $A^{(1)}, A^{(2)}, \ldots, A^{(N)}$, and $\mathcal{D}$ of Equation (1).

Output: Approximate solution $\mathcal{X}_m$ of (1).

1. Calculate the NKP matrices $Q_1, Q_2, \ldots, Q_N$ according to Proposition 5.1.
2. Compute $\widetilde{\mathcal{D}} = \mathcal{X} \times_1 Q_1^{-1} \times_2 \cdots \times_N Q_N^{-1}$, $\widetilde{\mathcal{B}}_n = \mathcal{X}_0 \times_1 Q_1^{-1} \times_2 \cdots \times_n Q_n^{-1} A^{(n)} \times_{n+1} \cdots \times_N Q_N^{-1}$.
3. Compute $\mathcal{R}_0 = \widetilde{\mathcal{D}} - (\widetilde{\mathcal{B}}_1 + \cdots + \widetilde{\mathcal{B}}_N)$, $\beta := \|\mathcal{R}_0\|$, and $\mathcal{V}_1 := \mathcal{R}_0/\beta$.
4. Define the $(m+1) \times m$ matrix $H_m = (h_{ij})$. Set $H_m = 0$.
5. For $j = 1, 2, \ldots, m$ do
6. Compute $\mathcal{W}_j := \mathcal{V}_j \times_1 Q_1^{-1} A^{(1)} \times_2 \cdots \times_N Q_N^{-1} + \cdots + \mathcal{V}_j \times_1 Q_1^{-1} \times_2 \cdots \times_N Q_N^{-1} A^{(N)}$.
7. For $i = 1, \ldots, j$ do
8. $h_{ij} := \langle \mathcal{W}_j, \mathcal{V}_i \rangle$
9. $\mathcal{W}_j := \mathcal{W}_j - h_{ij} \mathcal{V}_i$
10. End for
11. $h_{j+1,j} = \|\mathcal{W}_j\|$
12. $\mathcal{V}_{j+1} = \mathcal{W}_j / h_{j+1,j}$
13. End for
14. Compute $y_m = \mathrm{argmin}_y \|\beta e_1 - H_m y\|_2$ and $\mathcal{X}_m = \mathcal{X}_0 + \widetilde{\mathcal{V}} \bar{\times}_{N+1} y_m$, where $\widetilde{\mathcal{V}}$ is an $(N+1)$-mode tensor with column tensors $\mathcal{V}_1, \mathcal{V}_2, \ldots, \mathcal{V}_m$.
15. If satisfied, stop, otherwise, set $\mathcal{X}_0 := \mathcal{X}_m$ and go to 1.

6 Numerical experiments

In this section, we perform some numerical experiments to illustrate the effectiveness of our approach. All computations are implemented in MATLAB 2010a running on an Inter(R) Dual T3500 2.0 GHz notebook with 2 GB RAM, and all the implementations are based on the codes from the MATLAB Tensor Toolbox developed by Bader and Kolda [1, 2]. We use the zero vector (or zero tensor) as the starting point, and $\frac{\|r_k\|}{\|r_0\|} < \epsilon$ (or $\frac{\|\mathcal{R}_k\|}{\|\mathcal{R}_0\|} < \epsilon$) as the stopping condition for all of the iterative algorithms compared, where r_k (or $\mathcal{R}_k$) is the residual of the k-th iteration.

Example 6.1. In this example we compare the standard projection methods (SPM) with the projection methods based on the tensor format (PM_BTF) for solving (1). Let $I_n \equiv 20$, $\epsilon = 10^{-10}$. We vary the dimension N from 3 to 5. When $N = 3$, the coefficient matrices $A^{(n)}$ and the tensor $\mathcal{D}$ are generated (see [27], eg. 35.1) by the following MATLAB codes:

```
rand('state',0);
A1=eye(20)+0.5*randn(20)/sqrt(20);
A2=eye(20)+0.5*randn(20)/sqrt(20);
A3=eye(20)+0.5*randn(20)/sqrt(20);
D=tenrand([20,20,20]);
```

When $N = 4$ and 5, the coefficient matrices are analogous. The results are reported in Table 1, where "–" and `Ratio` stand for out of memory and ratio of the CPU times for SPM to the CPU times for PM_BTF, respectively.

From Table 1 we note that the effectiveness of PM_BTF are better than that of the SPM. Moreover, as N increases, the advantage of PM_BTF is more obvious.

Example 6.2. In this example, we compare DM_BSD with PM_BTF for solving (3). Since the DM_BSD is only suitable for $N = 3$, we fix $N = 3$, $\epsilon = 10^{-14}$ and vary the size I_n from 40 to 160. The coefficient matrices are generated as in Example 6.1. The results are given in Table 2.

From Table 2 we observe that DM_BSD is efficient for small sized matrices. However, as I_n increases, it becomes time-consuming, the BTF type methods are more efficient.

Table 1 Comparison of CPU times (in seconds) for SPM and PM_BTF

Methods	$N = 3$	$N = 4$	$N = 5$
GMRES(10)	0.390	9.422	–
GMRES(10)_BTF	0.312	6.349	100.5
`Ratio`	1.25	1.48	
BiCGSTAB	0.250	7.738	–
BiCGSTAB_BTF	0.203	3.260	57.48
`Ratio`	1.23	2.37	
CGS	0.265	6.646	–
CGS_BTF	0.234	2.902	56.91
`Ratio`	1.13	2.29	
TFQMR	0.358	8.065	–
TFQMR_BTF	0.249	3.791	63.49
`Ratio`	1.44	2.13	

Table 2 Comparison of CPU times (in seconds) for DM_BSD and PM_BTF when $N = 3$

Methods	$I_n = 40$	$I_n = 80$	$I_n = 120$	$I_n = 160$
DM_BSD	3.476	44.12	236.9	1169
GMRES(10)_BTF	3.588	28.43	80.57	195.5
BiCGSTAB_BTF	1.997	11.54	40.78	97.61
CGS_BTF	2.090	12.60	38.95	102.2
TFQMR_BTF	2.465	16.88	47.52	122.2

Example 6.3. This example is concerned with the effectiveness of the NKP preconditioner. Consider the convection-diffusion equation [4, 28],

$$\begin{aligned} &-\nu\Delta u + c^{\mathrm{T}}\nabla u = f \quad \text{in} \quad \Omega = [0,1]^N, \\ &u = 0 \quad \text{on} \quad \partial\Omega. \end{aligned}$$

A standard finite difference discretization on equidistant nodes, combined with a second order convergent scheme [10, 15] for the convection term leads to the linear system (9) with

$$A^{(n)} = \frac{\nu}{h^2}\begin{pmatrix} 2 & -1 & & & \\ -1 & 2 & -1 & & \\ & \ddots & \ddots & \ddots & \\ & & -1 & 2 & -1 \\ & & & -1 & 2 \end{pmatrix} + \frac{c_n}{4h}\begin{pmatrix} 3 & -5 & 1 & & \\ 1 & 3 & -5 & \ddots & \\ & \ddots & \ddots & \ddots & 1 \\ & & 1 & 3 & -5 \\ & & & 1 & 3 \end{pmatrix}, \quad n = 1, 2, \ldots, N.$$

Let $\nu = 0.0001$, $c_n = 10$ (see [28]). Firstly, we consider the spectral distribution of the original coefficient matrix and the preconditioned matrix. The results are shown in Figure 2.

From Figure 2 we observe that the spectrum of the preconditioned matrix clusters around 1. So we can expect the preconditioned equations to have better convergence behaviors. Now, let $N = 4$, $I_n = 20$ and the right hand side be uniformly distributed random numbers. We solve Equation (1) by GMRES(10) and BiCGSTAB. The results are presented in Figure 3 and Table 3.

In Figure 3 we see that with the NKP preconditioner the residuals of the projection methods are decreasing faster than those of the projection methods without the preconditioner. It can be seen from Table 3 that CPU times of the projection methods with the NKP preconditioner is less than that of the projection methods without the preconditioner. Note that the CPU times of the projection methods with the NKP preconditioner in Table 3 include the times for finding the NKP matrices, which usually

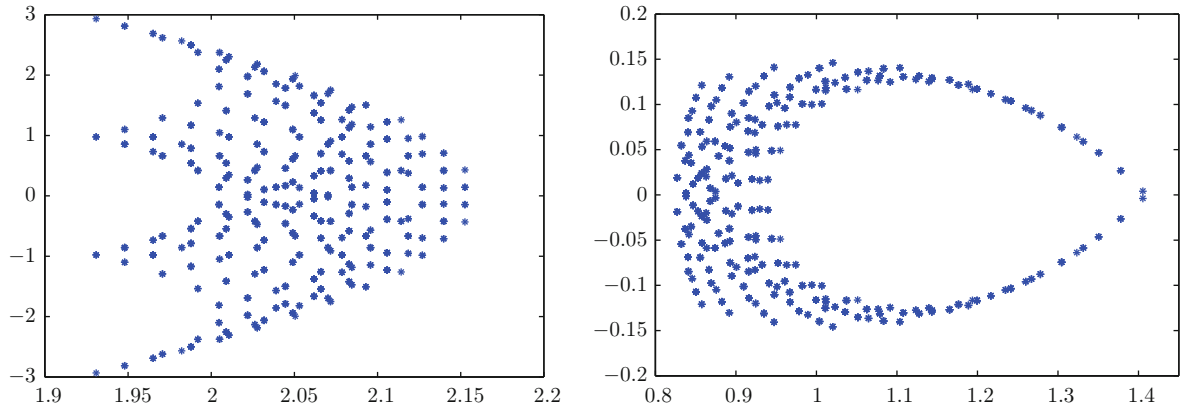


Figure 2 The spectral distributions of the original matrix (left) and the preconditioned matrix (right) for $I_n = 10, N = 3$

Table 3 CPU times (in seconds) for two iterative algorithms with none and the NKP preconditioner

Methods	10^{-6}	10^{-10}	10^{-14}
GMRES(10)_BTF	27.61	39.48	51.19
GMRES(10)_BTF+NKP	19.42	28.31	37.25
BiCGSTAB_BTF	26.75	30.81	34.30
BiCGSTAB_BTF+NKP	13.48	16.92	21.60

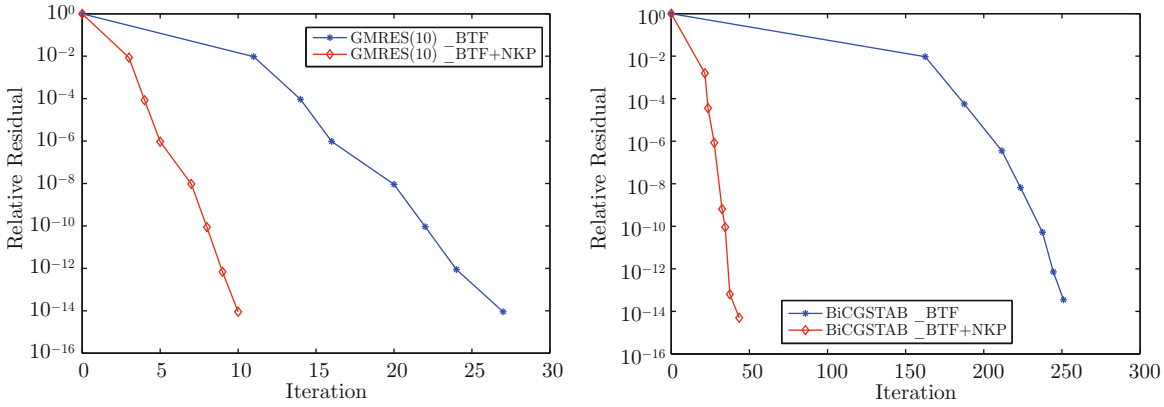


Figure 3 The convergence curves for two iterative algorithms with none and the NKP preconditioner

cost less than 1 second. Although the computational cost per iteration in the projection methods with the NKP preconditioner is larger than that of the projection methods without the preconditioner, the projection methods with the preconditioner are more efficient than those without the preconditioner.

7 Conclusions

In this paper, we propose the projection method based on the tensor format for solving (1). In contrast with the standard projection methods, our approach needs less flops and storage. Furthermore, we design a kind of NKP preconditioner to accelerate the convergence of the tensor format projection methods. The NKP preconditioner is easy to construct, since we only need to determine $2N$ parameters by using nonlinear optimization software. The NKP preconditioner is also storage-efficient, since we only need to store the NKP matrices instead of the preconditioner due to the use of Kronecker products.

Acknowledgements This work was supported by National Natural Science Foundation of China (Grant No. 10961010) and Science and Technology Foundation of Guizhou Province (Grant No. LKS[2009]03). We would like to thank Professor Ben-Wen Li at the key Laboratory of electromagnetic processing of materials of Northeastern University for his helpful suggestions and Professor Zheng-Jian Bai at School of Mathematical Sciences of Xiamen University for his constructive comments. We also cordially thank the referees for their valuable comments and suggestions that lead to the improvement of this paper.

References

1 Bader B W, Kolda T G. Efficient MATLAB computations with sparse and factored tensors. SIAM J Sci Comput, 2007, 30: 205–231
2 Bader B W, Kolda T G, MATLAB Tensor Toolbox, Version 2.4. Available at http://csmr.ca.sandia.gov/ ∼tgkolda/TensorToolbox/, 2010
3 Bai Z Z, Golub G H, Ng M K. Hermitian and skew-hermitian splitting methods for non-hermitian positive definite linear systems. SIAM J Matrix Anal Appl, 2003, 24: 603–626
4 Ballani J, Grasedyck L. A Projection method to solve linear systems in tensor format. Numer Linear Algebra Appl, doi: 10.1002/nla.1818
5 Beylkin G, Mohlenkamp M J. Algorithms for numerical analysis in high dimensions. SIAM J Sci Comput, 2005, 26: 2133–2159
6 Ding F, Chen T. Gradient based iterative algorithms for solving a class of matrix equations. IEEE Trans Automat Control, 2005, 50: 1216–1221
7 Ding F, Chen T. Iterative least squares solutions of coupled Sylvester matrix equations. Systems Control Lett, 2005, 54: 95–107
8 Golub G H, Nash S, Van Loan C F. A Hessenberg-Schur method for the problem $AX + XB = C$. IEEE Trans Auto Control, 1979, 24: 909–913

9 Golub G H, Van Loan C F. Matrix Computations, 3rd ed. Baltimore, Maryland: Johns Hopkins University Press, 1996
10 Grasedyck L. Existence and computation of low Kronecker-rank approximations for large linear systems of tensor product structure. Computing, 2004, 72: 247–265
11 Grasedyck L. Hierarchical singular value decomposition of tensors. SIAM J Matrix Anal Appl, 2010, 31: 2029–2054
12 Hu D Y, Reichel L. Krylov-subspace methods for the Sylvester equation. Linear Algebra Appl, 1992, 172: 283–313
13 Huyer W, Neumaier A. Global optimization by multilevel coordinate search. J Global Optim, 1999, 14: 331–355
14 Kolda T G, Bader B W. Tensor decompositions and applications. SIAM Rev, 2009, 51: 455–500
15 Kressner D, Tobler C. Krylov subspace methods for linear systems with tensor product structure. SIAM J Matrix Anal Appl, 2010, 31: 1688–1714
16 Langville A N, Stewart W J. A Kronecker product approximate preconditioner for SANs. Numer Linear Algebra Appl, 2004, 11: 723–752
17 Langville A N, Stewart W J. Testing the nearest Kronecker product preconditioner on Markov chains and stochastic automata networks. Informs J Comput, 2004, 16: 300–315
18 De Lathauwer L. A survey of tensor methods. In: Proceeding of the 2009 IEEE International Symposium on Circuits and Systems (ISCAS). Taipei: IEEE, 2009, 2773–2776
19 Li B W, Sun Y S, Zhang D W. Chebyshev collocation spectral methods for coupled radiation and conduction in a concentric spherical participating medium. ASME J Heat Transfer, 2009, 131: 062701–062709.
20 Li B W, Tian S, Sun Y S, et al. Schur-decomposition for 3D matrix equations and its application in solving radiative discrete ordinates equations discretized by Chebyshev collocation spectral method. J Comput Phys, 2010, 229: 1198–1212
21 Li J R, White J. Low-rank solutions of Lyapunov equations. SIAM J Matrix Anal Appl, 2002, 24: 260–280
22 Nagy J G, Kilmer M E. Kronecker product approximation for preconditioning in three-dimensional imaging applications. IEEE Transactions on Image Processing, 2006, 15: 604–613
23 Pitsianis N, Van Loan C F. Approximation with Kronecker products. In: Moonen M S, Golub G H, eds. Linear Algebra for Large Scale and Real Time Applications. Boston: Kluwer Academics, 1993, 293–314
24 Qi L, Sun W Y, Wang Y J. Numerical multilinear algebra and its applications. Front Math China, 2007, 2: 501–526
25 Saad Y. Iterative Methods for Sparse Linear Systems. Boston: PWS Publishing Company, 1996
26 Simoncini V. A new iterative method for solving large-scale Lyapunov matrix equations. SIAM J Sci Comput, 2007, 29: 1268–1288
27 Trefethen L N, Bau D. Numerical Linear Algebra. Philadelphia: SIAM, 1997
28 Xiang H, Grigori L. Kronecker product approximation preconditioners for convection-diffusion model problems, Numer Linear Algebra Appl, 2010, 17: 723–752