A Modified GMRES Method for Solving a Symmetric Solution to Lyapunov Equation for Multi-Agent Systems

Asuka Ohashi * and Kiyotsugu Takaba *

Abstract: We consider solving a large-scale Lyapunov equation for a multi-agent system. As is well known, the Lyapunov equation can be solved by equivalently rewriting it as a system of linear equations. The difficulties in solving this system are memory requirement and computational complexity due to the large-scale coefficient matrix involving a number of Kronecker products. This paper presents a modified GMRES method for solving the aforementioned system of linear equations taking account of its tensor structure and the symmetry of the unknown matrix in the Lyapunov equation. Through numerical experiments, the improvement in memory requirement and computational time by the present algorithms is verified in comparison with the previous GMRES-based methods.

Key Words: Lyapunov equation, symmetric solution, multi-agent system, tensor, n-mode product.

1. Introduction

We consider a Lyapunov equation for a linear multi-agent system distributed feedback control. The multi-agent system is a system that consists of a number of agents which interact with one another. Examples of such a system include vehicle formations, microgrids, and so on. It is well known that the collective dynamics of the system can be represented in terms of the Kronecker products (see, e.g. [1]).

A Lyapunov equation is one of the widely used tools to investigate the stability of linear systems. A conventional technique to solve the Lyapunov equation is to transform it into a system of linear equations using the Kronecker product (see, e.g. [2]). It is obvious that the size of the Lyapunov equation for the multi-agent system becomes very large according to the number of agents. In this case, the aforementioned transformation of the Lyapunov equation leads to a large-scale system of linear equations, which causes numerical difficulties such as huge memory requirement, computational time, etc.

In order to solve the system of linear equations with a large asymmetric sparse matrix, we focus on the generalized minimal residuals (GMRES) method [3]. The GMRES method is a well-known iterative method based on the Krylov subspace to monotonously decrease a residual norm. The advantage of the GMRES method is that the method requires only the results of matrix-vector multiplications of the coefficient matrix, i.e., the method does not require preserving the matrix. By using this advantage, the modified GMRES method [4] was proposed, where a third-order tensor and n-mode products played important roles. From the numerical experiments in [4], the modified GMRES method succeeded in reducing the required memory and runtime.

However, the modified GMRES method does not consider the symmetric structure of the unknown matrix in the Lyapunov equation. Based on the symmetry of the unknown matrix, we can reproduce the whole of it from solving a lower triangular part of it. In this paper, we will further improve the memory requirement of the modified method [4] by reducing the size of the aforementioned system of linear equations.

2. Collective Dynamics of Multi-Agents

In this section, we model the collective dynamics of the multi-agent system based on [5]. We focus on a simple case of the base as shown below. Let \( n \) be the number of agents in the group. The linear dynamics of the \( i \)-th agent is given by

\[
\dot{x}_i = Ax_i + Bu_i, \quad i \in \{1, 2, \ldots, n\},
\]

where \( A \in \mathbb{R}^{m \times m} \), \( B \in \mathbb{R}^{m \times p} \), and \( x_i(t) \in \mathbb{R}^m \) and \( u_i(t) \in \mathbb{R}^p \) are the agent state and control, respectively. Here, let \( N_i \) be the set of neighbors of the \( i \)-th agent. Then, the \( i \)-th agent receives the following measurements:

\[
y_i = C_1 x_i,
\]

\[
z_{ij} = C_2 (x_i - x_j),
\]

where \( C_1 \in \mathbb{R}^{k_{\text{in}}} \), \( C_2 \in \mathbb{R}^{k_{\text{out}}} \), \( j \in N_i \subset \{1, 2, \ldots, n\}\setminus\{i\} \), and \( y_i \in \mathbb{R}^{k_{\text{in}}} \) and \( z_{ij} \in \mathbb{R}^{k_{\text{out}}} \) represent the internal and external state measurements, respectively. Here, under the assumption of \( N_i \neq \emptyset \), the single signal error measurement is defined by

\[
z_i = \frac{1}{|N_i|} \sum_{j \in N_i} z_{ij},
\]

where \(|N_i|\) represents the cardinality of \( N_i \). Furthermore, the controls \( u_i \) are given by the static feedback laws

\[
u_i = D_1 y_i + D_2 z_i,
\]

where \( D_1 \in \mathbb{R}^{p \times k_{\text{in}}} \) and \( D_2 \in \mathbb{R}^{p \times k_{\text{out}}} \). Putting all the agent dynamics with the feedback control (5) together, we obtain

\[
\dot{x} = (I_n \otimes (A + BD_1 C_1)) + (I_n \otimes BD_2 C_2)(L \otimes I_m)x,
\]

where \( \otimes \) denotes the Kronecker product, which is a matrix multiplication that \( A \otimes B \in \mathbb{R}^{m \times p \times m \times p} \) for \( A \in \mathbb{R}^{m \times m} \) and \( B \in \mathbb{R}^{p \times p} \), and \( L \) is the normalized graph Laplacian defined by

\[\ldots\]
$L_{ii} = 1,$  
$L_{ij} = \begin{cases} 
\frac{1}{|M^i|}, & j \in N_i, \\
0, & j \notin N_i, 
\end{cases}$  

The size of the coefficient matrix in Eq. (6) is $mn \times mn$.

3. Lyapunov Equation of Eq. (6)

A Lyapunov equation for the system of Eq. (6) is given in this section. For simplicity, define $A' := A + BDTC_1$ and $B' := BDTC_2$ in Eq. (6). Then the Lyapunov equation is

$$P(I_m \otimes A' + L \otimes B') + (I_m \otimes A' + L \otimes B')^TP = -Q$$  

(9)

since $(I_m \otimes B')(L \otimes I_m) = L \otimes B'$ from a property of the Kronecker product. If the system (6) is stable, there is exactly one solution $P > 0$ for any $Q > 0$.

It is well known that the Lyapunov equation (9) can be rewritten as the system of linear equations

$$[(I_m \otimes (I_m \otimes A' + L \otimes B')) + [(I_m \otimes A' + L \otimes B') \otimes I_m]]vec(P) = -vec(Q)$$  

(10)

by using the vec operator defined in (A.4) with $(I, J, K) = (mn, mn, 1)$ and the identity $vec(SXT) = (T^T \otimes S)vec(X)$ [2]. Thus, we can obtain the solution $P$ to the Lyapunov equation by solving Eq. (10). Since the transformation of Eq. (9) into Eq. (10) increases the number of the Kronecker products, the size of the coefficient matrix of Eq. (10) is much larger than that of Eq. (9). In Sections 4 and 5, we will discuss the GMRES-based methods for solving Eq. (10).

4. The Previous GMRES-Based Methods

The conventional GMRES method for Eq. (10) is shown in Algorithm 1. The GMRES method was proposed in 1986 by Saad and Schultz [3], and several variants of the method have been proposed (see, e.g. [6],[7]). These methods are effective to solve a system of linear equations with a large asymmetric sparse coefficient matrix. The convergence of Algorithm 1 is monitored by the following residual norm:

$$r_i := \|H_{ij}y - [vec(Q)]_{i}e_i\|_2,$$  

(11)

where $e_i$ is the first canonical vector of the $(i + 1)$-dimensional space, and

$$H_{ik} := \begin{bmatrix} 
h_{1,1} & h_{1,2} & \cdots & h_{1,i} \\
h_{2,1} & h_{2,2} & \cdots & h_{2,i} \\
0 & 0 & \cdots & 0 \\
\vdots & \vdots & \ddots & \vdots \\
h_{i,j-1} & h_{i,j} & 0 & 0 \\
0 & 0 & \cdots & 0 
\end{bmatrix}.$$  

(12)

In implementing Algorithm 1, we require the $m^2n^2 \times m^2n^2$ matrix $[(I_m \otimes (I_m \otimes A' + L \otimes B')) + [(I_m \otimes A' + L \otimes B') \otimes I_m]]$ in the third line of Algorithm 1. The number of its elements is $m^2n^4$. Even if $m$ or $n$ increases a little, its memory requirement increases rapidly.

In the rest of this section, we describe an improvement of Algorithm 1 that proposed in [4] in terms of memory requirements using tensors. Tensors and their operations such as n-mode products and the vec operator are defined in Appendix A.

Decomposing the right-hand side of the third line of Algorithm 1 yields

$$\hat{v} := \hat{v}_1 + \hat{v}_2,$$  

(13)

where

$$\hat{v}_1 := (I_{mn} \otimes I_2 \otimes A' + I_{mn} \otimes L \otimes B')v_i,$$  

(14)

$$\hat{v}_2 := (I_m \otimes A' \otimes I_m + L \otimes B' \otimes I_m)v_i.$$  

(15)

We present a method to reduce memory requirements for computations of $\hat{v}_1$ and $\hat{v}_2$. First, we transform the vectors $\hat{v}_1$, $\hat{v}_2$, and $v_i$ in Eqs. (14) and (15) into tensors $\mathcal{V}_1$, $\mathcal{V}_2$, $\mathcal{V}_1^{(1)}$, and $\mathcal{V}_2^{(1)}$ via the reshape operator in Eq. (A.5). Moreover, by using Eq. (A.7), we transform matrix-vector multiplication of the matrices involving the Kronecker products into the n-mode products defined in Eqs. (A.1)–(A.3).

Next, the computation of $\hat{v}_1$ in Eq. (14) is below. From $\mathcal{V}_1^{(1)} := \text{reshape}(v_i, m, n, mn)$ and $\mathcal{V}_1 := \text{reshape}(\hat{v}_1, m, n, mn)$, it follows that

$$\mathcal{V}_1 := \mathcal{V}_1^{(1)} \times_1 A' + \mathcal{V}_1^{(1)} \times_1 B' \times_2 L.$$  

(16)

Similarly, the computation of $\hat{v}_2$ in Eq. (15) is as follows. Let $\mathcal{V}_2^{(2)} := \text{reshape}(v_i, mn, m, n)$ and $\mathcal{V}_2 := \text{reshape}(\hat{v}_2, mn, m, n)$. Then, we have

$$\mathcal{V}_2 := \mathcal{V}_2^{(2)} \times_2 A' + \mathcal{V}_2^{(2)} \times_2 B' \times_3 L.$$  

(17)

The computations of Eqs. (16) and (17) are summarized in Algorithm 2 [4].

It is easily noticed from the above discussion that Algorithms 1 and 2 are mathematically equivalent.

Algorithm 1 The GMRES method [3] for Eq. (10)

1: Set $v_1 := -vec(Q)|/||vec(Q)||$;
2: for $i = 1, 2, \ldots, k$ until convergence do
3: \quad $v_i := [(I_m \otimes (I_m \otimes A' + L \otimes B')) + [(I_m \otimes A' + L \otimes B') \otimes I_m]]v_{i-1}$;
4: \quad for $j = 1 : i$ do
5: \quad \quad $h_{ij} := v_{i}^T v_{j};$
6: \quad \quad $\hat{v} := \hat{v} - h_{ij}v_{j};$
7: \quad end for
8: \quad $r_i := ||H_{ij}y - [vec(Q)]_{i}e_i||_2$;
9: \quad if $r_i < \epsilon$ then
10: \quad \quad break;
11: \quad end if
12: end for
13: vec($P$) := [$v_1, v_2, \ldots, v_k]$;

Algorithm 2 The modified GMRES method [4] for Eq. (10)

1: Set $v_1 := -vec(Q)|/||vec(Q)||$;
2: for $i = 1, 2, \ldots, k$ until convergence do
3: \quad $\mathcal{V}_1^{(1)} := \text{reshape}(v_i, m, n, mn)$;
4: \quad $\mathcal{V}_1 := \mathcal{V}_1^{(1)} \times_1 A' + \mathcal{V}_1^{(1)} \times_1 B' \times_2 L$;
5: \quad $\mathcal{V}_2^{(2)} := \text{reshape}(v_i, mn, m, n)$;
6: \quad $\mathcal{V}_2 := \mathcal{V}_2^{(2)} \times_2 A' + \mathcal{V}_2^{(2)} \times_2 B' \times_3 L$;
7: \quad $\hat{v} := \text{vec($\mathcal{V}_1$)} + \text{vec($\mathcal{V}_2$)}$;
8: \quad for $j = 1 : i$ do
9: \quad \quad $h_{ij} := \hat{v}_i^T \hat{v}_j$;
10: \quad \quad $\hat{v} := \hat{v} - h_{ij}v_{j}$;
11: \quad end for
12: \quad $r_i := ||H_{ij}y - [vec(Q)]_{i}e_i||_2$;
13: \quad if $r_i < \epsilon$ then
14: \quad \quad break;
15: \quad end if
16: end for
17: vec($P$) := [$v_1, v_2, \ldots, v_k$];
5. Proposed Method

In this section, we focus on the structure of the unknown matrix $P$ in Eqs. (9) and (10). Since the unknown matrix $P$ is symmetric, we need only to solve for the lower triangular part of $P$:

$$ P = \begin{bmatrix} p_{12} & p_{13} & \cdots & p_{1m} \\ p_{21} & p_{22} & \cdots & p_{2m} \\ \vdots & \vdots & \ddots & \vdots \\ p_{m1} & p_{m2} & \cdots & p_{mm} \end{bmatrix} $$ (18)

Let $\tilde{p}_j$ be the $j$-th column vector of the lower triangular part of $P$ as in Eq. (18). Then combining $\tilde{p}_j$ ($j = 1, 2, \ldots, mn$), we have

$$ \tilde{p} = \begin{bmatrix} \tilde{p}_1 \\ \tilde{p}_2 \\ \vdots \\ \tilde{p}_{mn} \end{bmatrix} \in \mathbb{R}^\ell, \text{where } \ell := \frac{mn(mn+1)}{2}. $$ (19)

We can reproduce vec($P$) from $\tilde{p}$ via $P_1$ and can produce $\tilde{p}$ from vec($P$) via $P_2$ as

$$ P_1 \tilde{p} = \text{vec}(P), $$ (20)

$$ P_2^T \text{vec}(P) = \tilde{p}, $$ (21)

where $P_1, P_2 \in \mathbb{R}^{n^2 \times \ell}$ are the $mn \times mn$ block matrices defined by

$$ P_1(i, j) := \begin{cases} I_{mn-j+1} & (i = j, j \neq mn), \\ O_{i-1} & (i = j, j = mn), \\ e_{2}^{(mn-j+1)} & (j < i, i \neq mn), \\ O_{i-j} & (\text{otherwise}), \end{cases} $$ (22)

$$ P_2(i, j) := \begin{cases} I_{mn-j+1} & (i = j, j \neq mn), \\ O_i & (i = j, j = mn), \\ e_{2}^{(j+1)} & (j < i, i \neq mn), \\ O_{i-j} & (\text{otherwise}), \end{cases} $$ (23)

for $i, j = 1, 2, \ldots, mn$, and $e_{2}^{(j)}$ denotes the $j$-dimensional $i$-th row canonical vector.

For simplicity, define $T := [(I_{mn} \otimes (I_{n} \otimes A' + L \otimes B'))] + [(I_{n} \otimes (I_{m} \otimes A + L \otimes B') \otimes I_{nm})]$. Then, Eq. (10) is equivalently rewritten as the smaller system with the $\ell \times \ell$ coefficient matrix using $P_1$ and $P_2$:

$$ (P_1^T TP_1) \tilde{p} = -P_2^T \text{vec}(Q). $$ (24)

We propose Algorithm 4 by applying the same technique as Algorithm 2 to Eq. (24). For later comparison, we also present Algorithm 3 which uses the conventional GMRES method to solve Eq. (24). Similarly to Algorithms 1 and 2, Algorithms 3 and 4 are mathematically equivalent.

Algorithm 3 The GMRES method [3] for Eq. (24)

1: Set $v_1 = -P_2^T \text{vec}(Q) / \|P_2^T \text{vec}(Q)\|_2$
2: for $i = 1, 2, \ldots, k$ until convergence do
3: $v_i := P_2^T [(I_{mn} \otimes (I_{n} \otimes A' + L \otimes B')) + [(I_{n} \otimes (I_{m} \otimes A + L \otimes B') \otimes I_{nm})]v_{i-1}$
4: for $j = 1 : i$ do
5: $h_{ij} := v_i^T v_j$
6: end for
7: $h_{ij+1} := \|v_i\|_2$; $v_{i+1} := \hat{v} = h_{i+1,i}$
8: Find $y$ which minimizes $\|H_{ij} y - \|\text{vec}(Q)\| \|e_i\|_2$
9: $r_i := \|H_{ij} y - \|\text{vec}(Q)\| \|e_i\|_2$
10: if $r_i < \epsilon$ then
11: break;
12: end if
13: end for
14: $\tilde{p} := [v_1, v_2, \ldots, v_k]$
15: $P := \text{reshape}(P_1 \tilde{p}, mn, mn, 1)$

Algorithm 4 The modified GMRES method [4] for Eq. (24)

1: Set $v_1 = -P_2^T \text{vec}(Q) / \|P_2^T \text{vec}(Q)\|_2$
2: for $i = 1, 2, \ldots, k$ until convergence do
3: $v_i := P_2^T [(I_{mn} \otimes (I_{n} \otimes A' + L \otimes B')) + [(I_{n} \otimes (I_{m} \otimes A + L \otimes B') \otimes I_{nm})]v_{i-1}$
4: $v_i := \text{reshape}(v_i, m, n, mn)$
5: $v_i := \text{reshape}(v_i, n, m, mn)$
6: $v_i := \text{reshape}(v_i, m, n, n)$
7: $v_i := \text{reshape}(v_i, n, m, n)$
8: $v_i := \text{reshape}(v_i, m, n, n)$
9: for $j = 1 : i$ do
10: $h_{ij} := v_i^T v_j$
11: end for
12: $h_{ij+1} := \|v_i\|_2$; $v_{i+1} := \hat{v} = h_{i+1,i}$
13: Find $y$ which minimizes $\|H_{ij} y - \|\text{vec}(Q)\| \|e_i\|_2$
14: $r_i := \|H_{ij} y - \|\text{vec}(Q)\| \|e_i\|_2$
15: if $r_i < \epsilon$ then
16: break;
17: end if
18: end for
19: $\tilde{p} := [v_1, v_2, \ldots, v_k]$
20: $P := \text{reshape}(P_1 \tilde{p}, mn, mn, 1)$

6. Numerical Experiments

In this section, we report the results of numerical experiments using Algorithms 1–4. All computations are carried out using MATLAB version R2016a on a workstation with two 2.40 GHz Xeon processors and 32 GB memory running over the Windows 7 operating system.

We set the stopping criterion of Algorithms 1–4 to $r_i < 10^{-10}$ where $r_i$ is defined in Eq. (11). In Figs. 2–7 shown later, we plot the mean values of the results of 10 runs using each of the algorithms.

The test matrices used in this numerical experiment are generated as follows. Firstly, set $n := 5, 6, \ldots, 15$ and $m = 4, 6$ in Eq. (1). Secondly, we randomly generate each entry of the matrices $A'$ and $B'$ in Eq. (10) with the uniform distribution over the interval $(0, 1)$, and set $Q = I_{mn}$ in Eq. (10). Thirdly, to describe interactions among agents, we employ directed random graphs produced by a version of the small world network model due to Watts and Strogatz [8], whose average degrees and rewiring probabilities are set to round(n/3) and 1, respectively. The graph Laplacians $L$ in all the algorithms are obtained from the above graph structures. Note that we verify that all of the systems of linear equations with the obtained test matrices have the unique solutions by MATLAB function: l1yap.

In Fig. 1, the convergence histories are shown in the case of
$m = 4$ and $n = 10$. The figure shows that Algorithm 1 requires the most iterations among all the algorithms and that the residual histories of the others are almost the same. This is because Algorithm 1 is the most affected by the numerical errors due to the most multiplications in the matrix-vector multiplication of the largest matrix among all the algorithms.

Figures 2 and 3 show the numbers of iterations required for convergence. From the figures, similarly, for all numbers of agents $n$, Algorithm 1 requires the most iterations among all the algorithms. The numbers of iterations of Algorithm 2–4 are almost the same.

The required times are depicted in Figs. 4 and 5. It is easily seen that Algorithm 3 converges within the least time among all the algorithms. Since the coefficient matrix is generated at the beginning of Algorithm 3, each iteration requires less time than Algorithm 4.

Finally, the memory requirements are shown in Figs. 6 and 7. The figures show that Algorithms 3 and 4 require less memory than Algorithms 1 and 2 since the dimensions of the generated basis vectors at each iteration of Algorithms 3 and 4 are smaller than those of Algorithms 1 and 2.

7. Conclusion

In this paper, we have investigated the modified GMRES methods to solve the system of linear equations in Eq. (10) associated with the large-scale Lyapunov equation for the multi-agent system. Based on the symmetry of the unknown matrix, we have developed the new algorithm (Algorithm 4) to improve the memory and computational efficiencies of the previous method (Algorithm 2) [4] which is a version of modified GMRES method utilizing the $n$-mode products ($n = 1, 2, 3$). It has been verified from the numerical comparison of Algorithms 1–4 in the previous section that Algorithm 4 requires less memory and less computational time than Algorithm 2.

It remains as future work to further improve the memory requirement and convergence rate by combining the proposed method with the restarting techniques [3] and the preconditioning techniques [6], respectively.

References

[1] J.A. Fax and R.M. Murray: Information flow and cooperative control of vehicle formations, *IEEE Trans. Autom. Control,*
Appendix Multiplication and Operator of Tensors

In this appendix, the definitions of n-mode products and the vec operator for tensors are given. See, e.g. [9] for more details on n-mode products.

A tensor means a multidimensional array, and the n-mode product is a multiplication of a tensor and a matrix. In this paper, a third-order tensor is denoted by \( \mathbf{X}, \mathbf{Y}, \mathbf{P}, \) and \( \mathbf{Q} \), and the \((i, j, k)\)-th element of \( \mathbf{X} \in \mathbb{R}^{I \times J \times K} \) is denoted by \( x_{ijk} \) for \( i = 1, 2, \ldots, I \), \( j = 1, 2, \ldots, J \), and \( k = 1, 2, \ldots, K \).

The n-mode product \( x_n \) of a third-order tensor \( \mathbf{X} \in \mathbb{R}^{I \times J \times K} \) and a matrix has three types. First, the 1-mode product \( \mathbf{x}_1 \) of \( \mathbf{X} \) and \( \mathbf{A} \in \mathbb{R}^{P \times I} \) is defined as

\[
(\mathbf{X} \times_1 \mathbf{A})_{p,i} := \sum_{k=1}^{K} x_{ijk} a_{pk}, \quad (A.1)
\]

where \( a_{pq} \) denotes the \((p, q)\)-th element of \( \mathbf{A} \), \( i = 1, 2, \ldots, I \), \( j = 1, 2, \ldots, J \), and \( k = 1, 2, \ldots, K \). Second, the 2-mode product \( \mathbf{x}_2 \) of \( \mathbf{X} \) and \( \mathbf{B} \in \mathbb{R}^{J \times P} \) is defined as

\[
(\mathbf{X} \times_2 \mathbf{B})_{p,j} := \sum_{k=1}^{K} x_{ijk} b_{kp}, \quad (A.2)
\]

where \( b_{pq} \) denotes the \((p, q)\)-th element of \( \mathbf{B} \). Third, the 3-mode product \( \mathbf{x}_3 \) of \( \mathbf{X} \) and \( \mathbf{C} \in \mathbb{R}^{K \times P} \) is defined as

\[
(\mathbf{X} \times_3 \mathbf{C})_{p,k} := \sum_{j=1}^{J} x_{ijk} c_{pk}, \quad (A.3)
\]

where \( c_{pq} \) denotes the \((p, q)\)-th element of \( \mathbf{C} \).

The vec operator is an operator to vectorize a tensor by combining column vectors of the tensor into the vector:

\[
\text{vec} : \mathbb{R}^{I \times J \times K} \rightarrow \mathbb{R}^{IK}. \quad (A.4)
\]

In contrast, the reshape operator is an operator to reshape a vector into a third-order tensor whose dimensions are specified by \( I, J, \) and \( K \):

\[
\text{reshape} : \mathbb{R}^{IK} \rightarrow \mathbb{R}^{I \times J \times K}. \quad (A.5)
\]

The reshape operator is sometimes referred to as the \( \text{vec}^{-1} \) operator.

For example, the third-order tensor \( \mathbf{X} \in \mathbb{R}^{2 \times 3 \times 2} \) is represented as Fig. A.1.

\[
\mathbf{X} := \begin{bmatrix} x_{11} & x_{12} & x_{13} \\ x_{21} & x_{22} & x_{23} \\ x_{31} & x_{32} & x_{33} \\ x_{41} & x_{42} & x_{43} \end{bmatrix},
\]

Fig. A.1 Tensor \( \mathbf{X} \) with column vectors \( x_{ij} \).

Here, the matrices in \( \mathbf{X} \) are

\[
\begin{bmatrix} x_{1k} & x_{2k} & x_{3k} \end{bmatrix} = \begin{bmatrix} x_{11k} & x_{12k} & x_{13k} \\ x_{21k} & x_{22k} & x_{23k} \\ x_{31k} & x_{32k} & x_{33k} \\ x_{41k} & x_{42k} & x_{43k} \end{bmatrix},
\]

where \( k = 1, 2 \). Vectorizing the tensor \( \mathbf{X} \), we get

\[
\text{vec}(\mathbf{X}) = \begin{bmatrix} x_{11} \\ x_{12} \\ x_{13} \\ x_{21} \\ x_{22} \\ x_{23} \end{bmatrix} \in \mathbb{R}^{24}. \quad (A.6)
\]

From reshaping the vector in Eq. (A.6), that is, reshape(vec(\( \mathbf{X} \)), 2, 3, 4), \( \mathbf{X} \) is obtained.

Finally, from the definitions of the n-mode products in Eqs. (A.1)–(A.3), the vec operator, and the Kronecker product, the following identity holds true:

\[
(\mathbf{C} \otimes \mathbf{B} \otimes \mathbf{A})\text{vec}(\mathbf{X}) = \text{vec}(\mathbf{X} \times_1 \mathbf{A} \times_2 \mathbf{B} \times_3 \mathbf{C}), \quad (A.7)
\]

where \( \mathbf{X} \in \mathbb{R}^{I \times J \times K}, \mathbf{A} \in \mathbb{R}^{P_1 \times I}, \mathbf{B} \in \mathbb{R}^{P_2 \times J}, \) and \( \mathbf{C} \in \mathbb{R}^{P_3 \times K}.\)