Iterative Solution of Structured Problem

Tamara G. Kolda

February 3, 2026

1 Problem

Given a d-way tensor $\mathcal{T} \in \mathbb{R}^{n_1 \times n_2 \times \cdots \times n_d}$ such that the data is unaligned (meaning the tensor $\mathcal{T}$ has missing entries), we consider the problem of computing a CP decomposition of rank r where some modes are infinite-dimensional and constrained to be in a Reproducing Kernel Hilbert Space (RKHS). We want to solve this using an alternating optimization approach, and our question is focused on the mode-k subproblem for an infinite-dimensional mode. For the subproblem, then CP factor matrices $A_1, \ldots, A_{k-1}, A_{k+1}, \ldots, A_d$ are fixed, and we are solving for A_k.

Our notation is as follows. Let $N = \prod_i n_i$ denote the product of all sizes. Let $n \equiv n_k$ be the size of mode k, let $M = \prod_{i \neq k} n_i$ be the product of all dimensions except k, and assume $n \ll M$. Since the data are unaligned, this means only a subset of $\mathcal{T}$'s entries are observed, and we let $q \ll N$ denote the number of observed entries. We let $T \in \mathbb{R}^{n \times M}$ denote the mode-k unfolding of the tensor $\mathcal{T}$ with all missing entries set to zero. The vec operations creates a vector from a matrix by stacking its columns, and we let $S \in \mathbb{R}^{N \times q}$ denote the selection matrix (a subset of the $N \times N$ identity matrix) such that $S^T \operatorname{vec}(T)$ selects the q known entries of the tensor $\mathcal{T}$ from the vectorization of its mode-k unfolding. We let $Z = A_d \odot \cdots \odot A_{k+1} \odot A_{k-1} \odot \cdots \odot A_1 \in \mathbb{R}^{M \times r}$ be the Khatri-Rao product of the factor matrices corresponding to all modes except mode k. We let $B = TZ$ denote the MTTKRP of the tensor $\mathcal{T}$ and Khatri-Rao product Z.

We assume $A_k = KW$ where $K \in \mathbb{R}^{n \times n}$ denotes the psd RKHS kernel matrix for mode k. The matrix W of size $n \times r$ is the unknown for which we must solve. The system to be solved is

$$\left[(Z \otimes K)^T S S^T (Z \otimes K) + \lambda (I_r \otimes K)\right] \operatorname{vec}(W) = (I_r \otimes K) \operatorname{vec}(B). \tag{1}$$

Here, I_r denotes the $r \times r$ identity matrix. This is a system of size $nr \times nr$ Using a standard linear solver costs $O(n^3 r^3)$, and explicitly forming the matrix is an additional expense.

Explain how an iterative preconditioned conjugate gradient linear solver can be used to solve this problem more efficiently. Explain the method and choice of preconditioner. Explain in detail how the matrix-vector products are computed and why this works. Provide complexity analysis. We assume $n, r < q \ll N$. Avoid any computation of order N.

2 Context

This is an optimization problem that arises in fitting a canonical tensor decomposition to real-world data. Such techniques are often used for data exploration, finding correlations

among data elements (e.g., co-occurring gene expression), and data compression. The specific problem here is fitting a relatively novel version of tensor decomposition that allows some modes to be infinite-dimensional from a Reproducing Kernel Hilbert Space (RKHS); in other words, the decomposition is in terms of functions rather than vectors. The optimization problem reduces to a structured regression problem, and the goal of the question is to find an efficient iterative method to solve it.

3 Potential of Contamination

The work on this project began long before the "1st Proof" project was conceived. As a result, it has and is using various online systems that may have fed our data to AI models. We don't think this is the case, but the possibility exists. Specifically, the paper was written using Overleaf (thought not using their AI tools), various GitHub repositories contain the codes, and one author uses GitHub copilot for various tasks, mostly inline completion.

4 Solution Notes

What follows is a specific solution. As this is an open-ended question, there is the potential for many other possible solution. In particular, a relatively efficient solution is still possible without transforming the problem. There are a couple things to look for...

1. Is the problem reasonably and correctly transformed using the eigendecomposition of the kernel matrix in a way that leads to good preconditioning? (This is very advanced, and I would be impressed if the AI can do this.)

2. Does the solution include an explanation of how to do matrix-vector products efficiently? (This is relatively standard, and I think the AI should be able to solve this.)

3. Does the method create any objects of size $\prod_k n_k$ or $\prod_{i \neq k} n_i$? (This is bad because it would be even more expensive than the "standard solver" approach.)

4. Does it propose a reasonable preconditioner? (There are a lot of options here, depending on whether or not a transformation is used. The main criteria is that it must be easy to solve.)

Solution Source:
Johannes Brust and Tamara G. Kolda,
Fast and Accurate CP-HIFI Solution (tentative title), 2025.

We consider several approaches for solving Eq. (1) in the remainder of this section. We present a direct method for the symmetric linear system in Section 4.1, using an additional regularization term. In Section 4.2, we present a transformation of the symmetric system based on the eigendecomposition of K. In Section 4.4, we present an iterative method based on the transformed symmetric system, adding some regularization akin to the symmetric direct method. In Table 1 and Section 4.5, we provide an accounting of the costs and comparison of direct and iterative methods.

4.1 Direct Solution of UI Subproblem (Symmetric Form)

Equation (1) is an indefinite symmetric linear system of size $rn \times rn$. Since it is indefinite, we add a regularization term parameterized by $\rho > 0$ to ensure positive definiteness. The modified system is

$$[F^T F + \lambda(I_r \otimes K) + \rho I_{rn}] \operatorname{vec}(W) = \operatorname{vec}(KB), \tag{2}$$

where $F = S^T(Z \otimes UD)$. Observe that we have pulled K inside the vectorization on the right-hand side.

To compute F, we want to avoid forming the $N \times nr$ Kronecker product $Z \otimes K$ explicitly. Instead, we create two special matrices: $\hat{K} \in \mathbb{R}^{q \times n}$ and $\hat{Z} \in \mathbb{R}^{q \times r}$. Each index $\ell \in [q]$ corresponds to a known entry index that we denote as $(i_1^{(\ell)}, i_2^{(\ell)}, \ldots, i_d^{(\ell)}) \in \Omega$. Then, for each $\ell \in [q]$, we let

$$\hat{Z}(\ell,:) = \left(A_d(i_d^{(\ell)},:) * \cdots * A_{k+1}(i_{k+1}^{(\ell)},:) * A_{k-1}(i_{k-1}^{(\ell)},:) * \cdots * A_1(i_1^{(\ell)},:)\right)^T, \text{ and} \tag{3}$$

$$\hat{K}(\ell,:) = K(i_k^{(\ell)},:). \tag{4}$$

Here, $*$ represents elementwise multiplication. In other words, $\hat{Z}$ and $\hat{K}$ represent the subset of rows of Z and K, respectively, that corresponds to the known entries of $\mathcal{T}$. Then, row ℓ of F is given by

$$F(\ell,:) = \hat{Z}(\ell,:) \otimes \hat{K}(\ell,:). \tag{5}$$

4.2 Transforming the UI Subproblem

we can exploit a factorization of K to transform Eq. (1) into an equivalent but potentially better conditioned system. Assuming we have the eigendecomposition $K = UDU^T$, we can rewrite Eq. (1) by factoring out $(I_r \otimes U)$ to obtain

$$[\underbrace{(Z \otimes UD)^T S}_{\bar{F}^T} \underbrace{S^T(Z \otimes UD)}_{\bar{F}} + \lambda(I_r \otimes D)] \operatorname{vec}(\underbrace{U^T W}_{\bar{W}}) = \operatorname{vec}(\underbrace{DU^T B}_{\bar{B}}). \tag{6}$$

Now we have a transformed system in the variable $\bar{W} = U^T W$, and we can solve for W via $W = U\bar{W}$ after solving the system. Note that we cannot pull D into the definition of $\bar{W}$ because it is indefinite. We define $\bar{F} := S^T(Z \otimes UD) \in \mathbb{R}^{q \times rn}$, which is analogous to F with K replaced by UD. We define $\bar{B} := DU^T B \in \mathbb{R}^{n \times r}$. Adding a regularization term as before, we obtain the modified system

$$[\bar{F}^T \bar{F} + \lambda(I_r \otimes D) + \rho\, I_{rn}] \operatorname{vec}(\bar{W}) = \operatorname{vec}(\bar{B}). \tag{7}$$

4.3 Key Lemmas for PCG Solution of UI Subproblem

Before we continue to the details of solving Eq. (7) via PCG, we present some key lemmas about working with matrices where each row is a Kronecker product of rows of two other matrices. These lemmas are important for efficiently computing the matrix-vector products and a preconditioner needed for PCG. We state these generically here so they can be reused in other contexts.

Let $A \in \mathbb{R}^{q\times r}$ and $B \in \mathbb{R}^{q\times n}$. Define the $q \times rn$ matrix C row-wise as

$$C(\ell,:) = A(\ell,:) \otimes B(\ell,:), \quad \text{for} \quad \ell = 1,\ldots,q. \tag{8}$$

Recall that for the Kronecker product of an n-vector and an r-vector or the vectorization of an $n \times r$ matrix, there is a correspondence between $k \in [rn]$ and the pair (i,j) with $i \in [n]$ and $j \in [r]$ such that $k = i + (j-1)n$. For the Kronecker product means, this means $C_{\ell k} = B_{\ell i}A_{\ell j}$. For a vectorized matrix, we have $(\mathrm{vec}(X))_k = X_{ij}$.

Lemma 1 shows how to compute the matrix-vector product Cx efficiently. This would normally cost $\mathcal{O}(qrn)$ if we formed C explicitly. However, using the structure of C, we can compute it using only $\mathcal{O}(q(r+n))$ operations. Moreover, we avoid forming C explicitly, which reduces the memory from $\mathcal{O}(qrn)$ to $\mathcal{O}(q(r+n))$.

Lemma 1. *Given the setup in Eq.* (8)*, let $X \in \mathbb{R}^{n\times r}$ be a matrix and define $x = \mathrm{vec}(X)$. Then we have*

$$Cx = (A * BX)1_r.$$

Here 1_r denotes the r-vector of all ones.

Proof. For all $\ell = 1,\ldots,q$ we have

$$(Cx)_\ell = \sum_{k=1}^{rn} C_{\ell k}x_k = \sum_{j=1}^{n}\sum_{i=1}^{n} B_{\ell i}X_{ij}A_{\ell j} = \sum_{j=1}^{r}(BX)_{\ell j}A_{\ell j}. \qquad \square$$

Lemma 2 shows how to compute the matrix-vector product C^Tv without forming C explicitly. The cost is unchanged at $\mathcal{O}(qrn)$, but the memory is reduced from $\mathcal{O}(qrn)$ to $\mathcal{O}(q(r+n))$.

Lemma 2. *Given the setup in Eq.* (8)*, let $v \in \mathbb{R}^q$. Then we have*

$$C^Tv = \mathrm{vec}(B^T \,\mathrm{diag}(v)\, A).$$

Proof. Define $k = i + (j-1)n$ for $i = 1,\ldots,n$ and $j = 1,\ldots,r$. Then, we have

$$(C^Tv)_k = \sum_{\ell=1}^{q} C_{\ell k}v_\ell = \sum_{\ell=1}^{q} B_{\ell i}A_{\ell j}v_\ell = (B^T \,\mathrm{diag}(v)\, A)_{ij}. = \left(\mathrm{vec}(B^T \,\mathrm{diag}(v)\, A)\right)_k. \qquad \square$$

Lemma 3 shows how to compute the diagonal of C^TC efficiently. We reduce the computation from $\mathcal{O}(qr^2n^2)$ to $\mathcal{O}(q(r^2+n^2))$ operations. And, again, we avoid forming C explicitly, which reduces the memory from $\mathcal{O}(qrn)$ to $\mathcal{O}(q(r+n))$.

Lemma 3. *Given the setup in Eq.* (8)*. Then*

$$\mathrm{diag}(C^TC) = \mathrm{vec}\left((B*B)^T(A*A)\right).$$

Proof. Define $k = i + (j-1)n$ for $i = 1, \ldots, n$ and $j = 1, \ldots, r$. Then, we have

$$\begin{aligned}(C^TC)_{kk} &= \sum_{\ell=1}^{q} C_{\ell k}^2 = \sum_{\ell=1}^{q} B_{\ell i}^2 A_{\ell j}^2 \\ &= \left[(B * B)^T (A * A)\right]_{ij} = \left[\operatorname{vec}\left((B * B)^T (A * A)\right)\right]_k . \end{aligned}$$ □

We apply these results in the next section.

4.4 PCG Solution of Transformed UI Subproblem

We can form $\bar{F}$ similarly to how we formed F. We define $H = UD \in \mathbb{R}^{n \times n}$ and $\hat{H} \in \mathbb{R}^{q \times n}$ such that $\hat{H}(\ell, :) = H(i_k^{(\ell)}, :)$ for each $\ell \in [q]$. Then, for each $\ell \in [q]$, we let

$$\bar{F}(\ell, :) = \hat{Z}(\ell, :) \otimes \hat{H}(\ell, :). \tag{9}$$

Let $x \in \mathbb{R}^{rn}$ be an arbitrary vector, and let $X \in \mathbb{R}^{n \times r}$ be its matrix representation so that $\operatorname{vec}(X) = x$. From Lemmas 1 and 2 in Section 4.3, we can compute $\bar{F}^T \bar{F} x$ as $\operatorname{vec}\left(\hat{H}^T \operatorname{diag}\left((\hat{Z} * \hat{H}X)1_r\right) \hat{Z}\right)$.

Then, we can compute the matrix-vector products for the conjugate gradient iterations without forming any Kronecker products using

$$\left(\bar{F}^T \bar{F} + \lambda(I_r \otimes D) + \rho\, I_{rn}\right) x = \operatorname{vec}\left(\hat{H}^T \operatorname{diag}\left((\hat{Z} * \hat{H}X)1_r\right) \hat{Z} + \lambda DX + \rho X\right). \tag{10}$$

We propose a diagonal preconditioner of the form

$$\bar{D} = \operatorname{diag}(\operatorname{diag}(\bar{F}^T \bar{F})) + \lambda(I_r \otimes D) + \rho\, I_{rn}.$$

Observe that $\bar{d} := \operatorname{diag}(\bar{D})$ is easy to compute since

$$\begin{aligned}\bar{d} &= \operatorname{diag}(\operatorname{diag}(\operatorname{diag}(\bar{F}^T \bar{F})) + \lambda(I_r \otimes D) + \rho\, I_{rn}) \\ &= \operatorname{diag}(\bar{F}^T \bar{F}) + \lambda(1_r \otimes \operatorname{diag}(D)) + \rho\, 1_{rn} \\ &= \operatorname{vec}((\hat{H} * \hat{H})^T (\hat{Z} * \hat{Z})) + \lambda(1_r \otimes \operatorname{diag}(D)) + \rho\, 1_{rn}\end{aligned} \tag{11}$$

The last step comes from Lemma 3 in Section 4.3.

4.5 Comparison of Costs

A comparison of the direct solution of the original symmetric problem Eq. (2) and PCG iterative solutions of the transformed problem Eq. (7) are shown in Table 1. For PCG, we let p denote the number of iterations needed for convergence. Recall that d is the order of the tensor, n is the size of mode k, r is the target rank, and q is the number of known entries. In general, we do not make assumptions about the relative sizes of n and r. We do assume, however, that $d < n, r \ll q$. Because we are working with an incomplete tensors, the MTTKRP is relatively cheap and never dominates the cost.

Factorizing the kernel matrix K for the transformed system The eigendecomposition of K costs $\mathcal{O}(n^3)$ flops. We stress once again that this is only done *one time* before the outermost alternating optimization iterations begin. In the methods we compare here, this is needed only for the PCG iterative method.

Table 1: Comparison of costs to solve the mode-k unaligned infinite-dimensional subproblem Eq. (1) of size $nr \times nr$ where n is the size of mode k and r is the target tensor decomposition rank. The variable q is the number of known entries in the observed tensor $\mathcal{T}$. For the PCG iterative method, p is the number of iterations.

Description	Direct Symmetric	PCG Iterative
Factorize $K = UDU^T$ *one-time cost!*	—	$\mathcal{O}(n^3)$
Compute $\hat{Z}$ and MTTKRP $B := \mathcal{T}Z$	$\mathcal{O}(qrd)$	$\mathcal{O}(qrd)$
Form F (and G) or H	$\mathcal{O}(qrn)$	$\mathcal{O}(n^2)$
Form matrix for linear solve	$\mathcal{O}(qr^2n^2)$	—
Form right-hand side	$\mathcal{O}(n^2r)$	$\mathcal{O}(n^2r)$
Form Preconditioner ($\bar{d}$)	—	$\mathcal{O}(qn^2 + qr^2)$
Solve system	$\mathcal{O}(r^3n^3)$	$\mathcal{O}(pnqr)$
Recover W	—	$\mathcal{O}(n^2r)$
Total Cost	$\mathcal{O}(qn^2r^2 + n^3r^3)$	$\mathcal{O}(qn^2 + qr^2 + qnrp)$
Storage	$\mathcal{O}(qnr + r^2n^2)$	$\mathcal{O}(qn + qr)$

Shared costs of all methods The $q \times r$ matrix $\hat{Z}$ defined in Eq. (3) is used by both methods. Likewise, the $n \times r$ MTTKRP $B = TZ$ is used by all methods. The cost to compute $\hat{Z}$ is $\mathcal{O}(qrd)$, Computing B is an MTTKRP with an incomplete tensor (Ballard and Kolda, Tensor Decompositions for Data Science, Cambridge University Press, 2025 with PDF available freely online). This would normally cost $\mathcal{O}(qrd)$ operations, but we can use $\hat{Z}$ to reduce the cost to $\mathcal{O}(qr)$ operations.

Direct solve of symmetric regularized system We first analyze the cost to form and solve the system as discussed in Section 4.1. We have to explicitly form F to form the system in Eq. (2). The cost to compute the $q \times rn$ matrix F is $\mathcal{O}(qrn)$ and requires $\mathcal{O}(qrn)$ storage. Forming the $rn \times rn$ matrix $F'F + \lambda(I_r \otimes K) + \rho\, I_{rn}$ is dominated by the cost to compute $F'F$, which costs $\mathcal{O}(qr^2n^2)$ operations. We also have to compute the right-hand side $\text{vec}(KB)$, which costs $\mathcal{O}(n^2r)$ operations. Finally, using a direct method such as Cholesky to solve the system costs $\mathcal{O}((rn)^3)$ operations. The storage is either dominated by storing F or the system matrix, which is $\mathcal{O}(rnq + r^2n^2)$.

PCG iterative solve of transformed system We now analyze the cost of using PCG to solve the transformed system Eq. (7) as discussed in Section 4.4. The right hand side $\text{vec}(\bar{B}) = \text{vec}(DU^TB)$ can be computed at a cost of $\mathcal{O}(n^2r)$ operations. We first have to compute the $n \times n$ matrix $H := UD$, which costs $\mathcal{O}(n^2)$ operations. Forming the diagonal preconditioner, the rn-vector $\bar{d}$ in Eq. (11), costs $\mathcal{O}(qn^2 + qr^2)$ operations. We never form $\bar{F}$ explicitly, which saves both computation and storage. Each matrix vector product is computed as in Eq. (10) at a cost of $\mathcal{O}(qnr)$ operations. Each preconditioner application costs $\mathcal{O}(rn)$ operations. Assuming that PCG converges in p iterations, the total cost for the PCG iterations is $\mathcal{O}(pqnr)$ operations. Finally, after solving for $\bar{W}$, we have to recover $W = U\bar{W}$, which costs $\mathcal{O}(n^2r)$ operations. The storage needed for PCG is dominated by storing $\hat{Z}$ and $\hat{H}$, which is $\mathcal{O}(qn + qr)$.

Summary and Comparison The direct method is cubic in the size of the unknown matrix W. In contrast, the PCG iterative method has a cost that is orders of magnitude lower, depending on the number of iterations p needed for convergence and the relative sizes of n, r, and p. In general, we expect the problem to be well conditioned so that p is not too large. The PCG method also has significantly lower storage requirements. Assuming $r < n < rn < q$, we have qrn storage for the direct methods versus qn storage for PCG.