The prompts and responses below have been edited only for formatting compatibility (to compile in a LaTeXdocument).

Fullproof # 10

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

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

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.

Model Response (Agent B)

Problem Formulation

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 incomplete (meaning the tensor $\mathcal{T}$ has missing entries), we consider the problem of computing a Canonical Polyadic (CP) decomposition of rank r. We assume that the true underlying generative functions for some modes are continuous and reside in an infinite-dimensional Reproducing Kernel Hilbert Space (RKHS), whereas the observed tensor $\mathcal{T}$ represents a finite-dimensional discretization (or evaluation) of these functions at discrete points.

We solve this decomposition problem using an alternating optimization approach. Our analysis focuses on the mode-k subproblem for such an RKHS-constrained mode. For the subproblem, the CP factor matrices $A_1, \ldots, A_{k-1}, A_{k+1}, \ldots, A_d$ are fixed, and we are solving for the factor matrix A_k.

Notation and Model Setup

Let $N = \prod_{i=1}^{d} n_i$ denote the product of all tensor dimensions. Let $n \equiv n_k$ be the size of mode k, and let $M = \prod_{i \neq k} n_i$ be the product of all dimensions except k, where we assume $n \ll M$. Since the data is incomplete, only a subset of $\mathcal{T}$'s entries are observed. We let $q \ll N$ denote the number of observed entries. Let $T \in \mathbb{R}^{n \times M}$ denote the mode-k unfolding of the tensor $\mathcal{T}$ with all missing entries initialized to zero.

The $\operatorname{vec}(\cdot)$ operation creates a vector from a matrix by stacking its columns. We let $S \in \mathbb{R}^{N \times q}$ denote the selection matrix (composed of a subset of the columns 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.

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. To strictly avoid $\mathcal{O}(N)$ memory and computational limits, Z is never explicitly constructed in full. We let $B = TZ \in \mathbb{R}^{n \times r}$ denote the Matricized Tensor Times Khatri-Rao Product (MTTKRP) of the tensor $\mathcal{T}$ and the conceptually defined Khatri-Rao product Z.

We parameterize the factor matrix as $A_k = KW$, where $K \in \mathbb{R}^{n \times n}$ denotes the symmetric, positive semi-definite RKHS kernel matrix for mode k evaluated at the discretization points. The matrix $W \in \mathbb{R}^{n \times r}$ is the unknown variable for which we must solve.

Derivation of the Linear System

To find W, we minimize a regularized least-squares objective function over the observed entries. Setting the gradient of this objective with respect to $\operatorname{vec}(W)$ to zero yields a linear system. The right-hand side (RHS) of this system stems from the gradient of the data fidelity term, which initially takes the form $(Z \otimes K)^T S S^T \operatorname{vec}(T)$. We rigorously simplify this expression. Since the missing entries in T are initialized to zero, applying the projection matrix SS^T acts identically on the non-zero entries, yielding $SS^T \operatorname{vec}(T) = \operatorname{vec}(T)$. Consequently, the RHS evaluates as:

$$\begin{aligned}(Z \otimes K)^T S S^T \operatorname{vec}(T) &= (Z \otimes K)^T \operatorname{vec}(T) \\ &= (Z^T \otimes K^T) \operatorname{vec}(T).\end{aligned}$$

Because the kernel matrix K is symmetric ($K^T = K$), standard properties of the

Kronecker product allow us to rewrite this cleanly as:

$$(Z^T \otimes K)\,\mathrm{vec}(T) = \mathrm{vec}(KTZ).$$

Substituting the MTTKRP definition $B = TZ$, we arrive at $\mathrm{vec}(KB) = (I_r \otimes K)\,\mathrm{vec}(B)$.
Thus, the linear system to be solved is:

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

where I_r denotes the $r \times r$ identity matrix and $\lambda > 0$ is the regularization parameter. This is a symmetric positive semi-definite system of size $nr \times nr$. Using a standard direct linear solver requires $\mathcal{O}(n^3 r^3)$ operations. Furthermore, explicitly forming the system matrix entails an additional and prohibitive computational expense. Given the operational assumption that $n, r < q \ll N$, it is imperative to solve this system without performing any computations of order N.

Preconditioned Conjugate Gradient Solver

To efficiently solve the large-scale system in (1), we employ the Preconditioned Conjugate Gradient (PCG) method. As an iterative algorithm, PCG requires only the action of the system matrix on a vector, bypassing the need to explicitly construct the $nr \times nr$ coefficient matrix.

Efficient Matrix-Vector Products

The core computational step in PCG is evaluating the matrix-vector product $y = \mathcal{H}\,\mathrm{vec}(V)$ at each iteration, where $\mathcal{H} = (Z \otimes K)^T SS^T (Z \otimes K) + \lambda(I_r \otimes K)$ is the system matrix, and $V \in \mathbb{R}^{n \times r}$ is a reshaped intermediate dense matrix representing the search direction.
We evaluate the product y in an $\mathcal{O}(N)$-free manner through the following sequence of operations:

1. **First Kernel Multiplication:** Compute $U = KV$. Since $K \in \mathbb{R}^{n \times n}$ and $V \in \mathbb{R}^{n \times r}$, this dense matrix multiplication requires $\mathcal{O}(n^2 r)$ operations.

2. **Sparse Residual Evaluation:** We observe that $(Z \otimes K)\,\mathrm{vec}(V) = \mathrm{vec}(KVZ^T) = \mathrm{vec}(UZ^T)$. The operation $SS^T\,\mathrm{vec}(UZ^T)$ extracts the entries of the dense $n \times M$ matrix UZ^T strictly at the q observed indices specified by S, effectively mapping all unobserved entries to zero.

   Instead of computing the full dense $n \times M$ matrix UZ^T, we evaluate the inner products of the corresponding rows of U and Z exclusively for the q observed entries. Crucially, to strictly avoid an $\mathcal{O}(N)$ memory and time bottleneck, the full Khatri-Rao product matrix $Z \in \mathbb{R}^{M \times r}$ is never explicitly formed. The required rows of Z (corresponding to the multi-indices of the q observed entries) are evaluated *on-the-fly* from the underlying CP factor matrices A_i ($i \neq k$). Assuming the tensor order d is a small constant, forming a single row of Z

takes $\mathcal{O}(r)$ operations. This dynamically generates a sparse $n \times M$ matrix Y containing exactly q non-zero elements. Generating the necessary rows of Z dynamically and evaluating these q specific entries requires strictly $\mathcal{O}(qr)$ operations.

3. **Sparse Matrix Multiplication:** Next, leveraging the symmetry of K ($K^T = K$), we apply the transpose operator $(Z \otimes K)^T = Z^T \otimes K^T = Z^T \otimes K$ to the sparse vector $\operatorname{vec}(Y)$. By the standard Kronecker product vectorization identity $\operatorname{vec}(ABC) = (C^T \otimes A)\operatorname{vec}(B)$, applying this operator to $\operatorname{vec}(Y)$ translates to the mathematical operation $\operatorname{vec}(KYZ)$.

   To compute this efficiently, we first evaluate the intermediate unvectorized product $P = YZ \in \mathbb{R}^{n\times r}$. Because Y possesses only q non-zeros, its right-multiplication only requires the rows of Z corresponding to its non-zero columns. Rather than instantiating Z as a full matrix, we employ standard sparse MTTKRP techniques: we dynamically evaluate only these necessary rows of Z on-the-fly (or reuse them if cached from the previous step). This sparse matrix multiplication cleanly circumvents the $\mathcal{O}(N)$ instantiation of Z and requires exactly $\mathcal{O}(qr)$ operations.

4. **Second Kernel Multiplication:** We left-multiply P by K to compute $KP \in \mathbb{R}^{n\times r}$. This dense matrix multiplication requires $\mathcal{O}(n^2 r)$ operations.

5. **Regularization Addition:** The Tikhonov regularization term simplifies to $\lambda(I_r \otimes K)\operatorname{vec}(V) = \operatorname{vec}(\lambda KV) = \operatorname{vec}(\lambda U)$.

The final evaluated vector y is cleanly obtained via the addition:

$$y = \operatorname{vec}(KP + \lambda U).$$

Jacobi Preconditioner

The convergence rate of PCG strongly depends on the condition number of the system matrix. RKHS kernel matrices frequently exhibit rapidly decaying eigenvalues, and the non-uniform sampling induced by SS^T further degrades system conditioning. To accelerate convergence, we utilize a Jacobi (diagonal) preconditioner $\mathcal{M}$, defined as the inverse of the diagonal elements of $\mathcal{H}$.

The diagonal elements of $\mathcal{H}$ can be extracted efficiently without forming the full matrix. Let $\tilde{K} = K \circ K \in \mathbb{R}^{n\times n}$, where $\circ$ denotes the Hadamard (element-wise) product. Conceptually, let $\tilde{Z} = Z \circ Z \in \mathbb{R}^{M\times r}$ denote the element-wise square of the Khatri-Rao product. Let $\Omega \in \mathbb{R}^{n\times M}$ be the sparse binary indicator matrix of the observed entries, such that $\Omega_{i,j} = 1$ if the entry (i, j) is observed, and 0 otherwise.

We now rigorously derive the diagonal of the data-fidelity term $(Z \otimes K)^T SS^T (Z \otimes K)$. For a generic matrix $Q = Z \otimes K$ and a diagonal weight matrix $\mathcal{W} = SS^T$, the diagonal of the quadratic form $Q^T \mathcal{W} Q$ (expressed as a column vector) is given by the identity $(Q \circ Q)^T \operatorname{diag}(\mathcal{W})$. Since the diagonal of the projection matrix SS^T evaluates exactly to the vectorized observation mask $\operatorname{vec}(\Omega)$, we can expand this expression using the mixed-product property of the Kronecker and Hadamard products. Letting

$\mathrm{vec}(D_{\mathrm{data}})$ denote the diagonal of the data-fidelity term reshaped as a vector, we have:

$$\begin{aligned}
\mathrm{vec}(D_{\mathrm{data}}) &= ((Z \otimes K) \circ (Z \otimes K))^T \,\mathrm{vec}(\Omega) \\
&= ((Z \circ Z) \otimes (K \circ K))^T \,\mathrm{vec}(\Omega) \\
&= (\tilde{Z}^T \otimes \tilde{K}^T) \,\mathrm{vec}(\Omega).
\end{aligned}$$

Because the kernel matrix K is symmetric, its Hadamard square $\tilde{K}$ is also symmetric ($\tilde{K}^T = \tilde{K}$). Thus, the expression simplifies to $(\tilde{Z}^T \otimes \tilde{K}) \,\mathrm{vec}(\Omega)$.
Finally, utilizing the standard Kronecker product vectorization identity $(B^T \otimes A)\,\mathrm{vec}(X) = \mathrm{vec}(AXB)$, we substitute $B = \tilde{Z}$, $A = \tilde{K}$, and $X = \Omega$ to directly evaluate the expression as $\mathrm{vec}(\tilde{K}\Omega\tilde{Z})$. Reshaping this vector back into an $n \times r$ matrix rigorously yields the stated analytical form:

$$D_{\mathrm{data}} = \tilde{K}(\Omega\tilde{Z}).$$

To preserve the $\mathcal{O}(N)$-free complexity bound, the matrix $\tilde{Z}$ is never fully formed. Instead, the product $\Omega\tilde{Z}$ is evaluated by computing only the required rows of $\tilde{Z}$ corresponding to the non-zero columns of Ω. These rows are obtained on-the-fly by squaring the elements of the dynamically computed rows of Z. Computing the sparse-dense matrix product $\Omega\tilde{Z}$ in this dynamic fashion costs $\mathcal{O}(qr)$, and the subsequent dense multiplication by $\tilde{K}$ costs $\mathcal{O}(n^2 r)$.
The diagonal of the regularization term $\lambda(I_r \otimes K)$, when reshaped into an $n \times r$ matrix, is simply $D_{\mathrm{reg}} = \lambda \,\mathrm{diag}(K)\mathbf{1}_r^T$, where $\mathrm{diag}(K)$ is the column vector containing the diagonal entries of K, and $\mathbf{1}_r \in \mathbb{R}^r$ is the vector of all ones.
Thus, the Jacobi preconditioner is formed by taking the element-wise inverse of $D = D_{\mathrm{data}} + D_{\mathrm{reg}}$. Constructing this preconditioner upfront requires a one-time cost of $\mathcal{O}(qr + n^2 r)$.

Complexity Analysis

By meticulously structuring the computations as outlined above and leveraging the on-the-fly row evaluation of Khatri-Rao products, the evaluation of the matrix-vector products and the preconditioner rigorously circumvents the prohibitive $\mathcal{O}(N)$ operations that would result from forming the full Khatri-Rao product or explicitly processing the full unfolded tensor. The overall computational complexity per PCG iteration is strictly bounded by the sum of the individual algorithmic steps:

$$\mathcal{O}(n^2 r) + \mathcal{O}(qr) + \mathcal{O}(qr) + \mathcal{O}(n^2 r) + \mathcal{O}(nr) = \mathcal{O}(qr + n^2 r).$$

Given the operational regime where $n, r < q \ll N$, this strategy guarantees a massive reduction in computational overhead, establishing a highly scalable and optimal method for updating infinite-dimensional RKHS modes within large-scale, incomplete CP tensor decompositions.