PCG for the infinite-dimensional mode-k subproblem (missing data)

We consider the linear system in the unknown $W \in \mathbb{R}^{n\times r}$

$$\Big[(Z \otimes K)^\top SS^\top (Z \otimes K) + \lambda(I_r \otimes K)\Big] \operatorname{vec}(W) = (I_r \otimes K) \operatorname{vec}(B), \tag{1}$$

where $K \in \mathbb{R}^{n\times n}$ is a (symmetric) psd kernel matrix, $Z \in \mathbb{R}^{M\times r}$ is the Khatri–Rao product of the other factors, $S \in \mathbb{R}^{N\times q}$ selects the q observed entries (so $S^\top \operatorname{vec}(T)$ equals the observed values), and $B = TZ \in \mathbb{R}^{n\times r}$. Throughout we assume $n, r < q \ll N = nM$ and avoid any $O(N)$ work.

1. Variational form, symmetry, and positive definiteness. Let $P \equiv SS^\top \in \mathbb{R}^{N\times N}$ be the diagonal "mask" matrix that keeps observed entries and zeros missing ones (so $P = P^\top = P^2$). The system (1) is the normal equation for the regularized least-squares objective

$$\min_{W\in\mathbb{R}^{n\times r}} \frac{1}{2}\left\|S^\top \operatorname{vec}(T) - S^\top \operatorname{vec}(KWZ^\top)\right\|_2^2 + \frac{\lambda}{2}\operatorname{Tr}(W^\top KW) \;=\; \frac{1}{2}\|P\circ(T - KWZ^\top)\|_F^2 + \frac{\lambda}{2}\operatorname{Tr}(W^\top KW), \tag{2}$$

where $\circ$ denotes Hadamard product and we used $\operatorname{Tr}(W^\top KW) = \|W\|_{\mathcal{H}}^2$ as the RKHS penalty. Writing the first term as $\frac{1}{2}\|P^{1/2}(\operatorname{vec}(T) - (Z\otimes K)\operatorname{vec}(W))\|_2^2$ shows the Hessian is

$$A \equiv (Z\otimes K)^\top P (Z\otimes K) + \lambda(I_r \otimes K) \in \mathbb{R}^{nr\times nr}, \qquad b \equiv (I_r\otimes K)\operatorname{vec}(B) = \operatorname{vec}(KB).$$

Thus A is symmetric. If $K \succ 0$ and $\lambda > 0$, then $A \succ 0$ because for any $x \neq 0$,

$$x^\top Ax = \|P^{1/2}(Z\otimes K)x\|_2^2 + \lambda\, x^\top (I_r\otimes K)x \geq \lambda\, x^\top(I_r\otimes K)x > 0.$$

(If K is only psd, one can add a nugget εI_n to K or instead regularize with $\lambda(I_r \otimes I_n)$. Alternatively, write $K = U\Lambda U^\top$ with rank m and parameterize $A_k = KW = U\Lambda\widetilde{W}$, reducing the unknown to $\widetilde{W} \in \mathbb{R}^{m\times r}$ and yielding an SPD system of size mr.) Hence we can solve (1) with (preconditioned) conjugate gradients (CG/PCG).

2. Why PCG helps. Direct solution would require forming A and performing a dense factorization costing $O((nr)^3) = O(n^3r^3)$. In contrast, PCG requires only: (i) repeated matrix–vector products $y \leftarrow Ax$ and (ii) repeated applications of a preconditioner M^{-1}, with overall cost $\approx$ #iters $\times$ (matvec + precond). Our goal is to implement both in $O(n^2r + qr)$ time per iteration and memory $O(nr + qr)$, never touching N-scale arrays.

3. PCG (brief). Choose an SPD preconditioner $M \approx A$ that is cheap to invert. Starting from x_0 (often 0 or the previous ALS iterate), define $r_0 = b - Ax_0$ and solve $Mz_0 = r_0$. Set $p_0 = z_0$ and for $t = 0, 1, 2, \ldots$ iterate

$$\alpha_t = \frac{\langle r_t, z_t\rangle}{\langle p_t, Ap_t\rangle}, \quad x_{t+1} = x_t + \alpha_t p_t, \quad r_{t+1} = r_t - \alpha_t Ap_t, \quad \text{solve } Mz_{t+1} = r_{t+1}, \quad \beta_t = \frac{\langle r_{t+1}, z_{t+1}\rangle}{\langle r_t, z_t\rangle}, \quad p_{t+1} = z_{t+1} + \beta_t$$

The algorithm only needs the ability to compute Ap_t (matvec) and to apply M^{-1}.

Efficient matrix–vector products without forming A

Operator viewpoint. Define the linear map $\mathcal{L} : \mathbb{R}^{n\times r} \to \mathbb{R}^q$ by

$$(\mathcal{L}(X))_t \equiv (KXZ^\top)_{i_t, j_t}, \qquad t = 1, \ldots, q.$$

Then the data term in (2) is $\frac{1}{2}\|\mathcal{L}(W) - y\|_2^2$ with $y \equiv S^\top \operatorname{vec}(T)$, and the normal equation is

$$(\mathcal{L}^\top\mathcal{L} + \lambda\,\mathcal{R})\operatorname{vec}(W) = \mathcal{L}^\top y, \qquad \mathcal{R} = I_r \otimes K.$$

Adjoint $\mathcal{L}^\top$ (scatter–accumulate formula). Equip $\mathbb{R}^q$ and $\mathbb{R}^{n\times r}$ with the Euclidean and Frobenius inner products. Given $u \in \mathbb{R}^q$, let $U \in \mathbb{R}^{n\times M}$ be the sparse matrix with $(U)_{i_t,j_t} = u_t$ (all other entries 0), i.e. $\operatorname{vec}(U) = Su$. Then

$$\mathcal{L}^\top u = \operatorname{vec}\big(K\,U\,Z\big) \in \mathbb{R}^{nr}. \tag{3}$$

Proof. For any $X \in \mathbb{R}^{n\times r}$,

$$\langle \mathcal{L}(X), u\rangle = \sum_{t=1}^{q} u_t (KXZ^\top)_{i_t,j_t} = \langle U, KXZ^\top\rangle_F = \langle KUZ, X\rangle_F = \langle \operatorname{vec}(KUZ), \operatorname{vec}(X)\rangle.$$

Hence $\mathcal{L}^\top u = \operatorname{vec}(KUZ)$.

Crucially, neither U nor Z need be formed: $UZ \in \mathbb{R}^{n\times r}$ is computed by accumulating the q nonzeros as in (8), and multiplying by K costs $O(n^2 r)$. Thus both $\mathcal{L}(X)$ (gather) and $\mathcal{L}^\top u$ (scatter/accumulate then multiply by K) can be applied in $O(qr + n^2 r)$ time.

Represent an input vector $x \in \mathbb{R}^{nr}$ as a matrix $X \in \mathbb{R}^{n\times r}$ such that $x = \operatorname{vec}(X)$. Use the identity

$$(Z \otimes K)\operatorname{vec}(X) = \operatorname{vec}(KXZ^\top), \tag{4}$$

which is a special case of $\operatorname{vec}(AXB^\top) = (B\otimes A)\operatorname{vec}(X)$.

Observed-entry operator implemented with index lists

Let the observed entries of the mode-k unfolding be indexed by pairs (i_t, j_t) for $t = 1,\dots,q$ with $i_t \in [n]$ and $j_t \in [M]$. Then for any matrix $U \in \mathbb{R}^{n\times M}$,

$$S^\top \operatorname{vec}(U) = \big(U_{i_t,j_t}\big)_{t=1}^{q} \in \mathbb{R}^q, \qquad \text{and} \qquad \operatorname{reshape}_{n\times M}(Sv) \text{ has nonzeros } (i_t, j_t) \text{ equal to } v_t.$$

Thus we can realize $S^\top$ (gather) and S (scatter) in $O(q)$ time using stored index arrays (i_t, j_t).

Avoiding explicit formation of Z. Although $Z \in \mathbb{R}^{M\times r}$ is defined as a Khatri–Rao product, $M = \prod_{i\neq k} n_i$ can be enormous, so we do *not* store Z. Instead, for each observed tensor entry we typically store its full multi-index $(i_1^{(t)},\dots,i_d^{(t)})$; the corresponding row needed in (5) and (8) is

$$Z_{j_t,:} \;=\; A_d(i_d^{(t)},:) \odot \cdots \odot A_{k+1}(i_{k+1}^{(t)},:) \odot A_{k-1}(i_{k-1}^{(t)},:) \odot \cdots \odot A_1(i_1^{(t)},:),$$

which can be computed on the fly in $O((d-1)r)$ time per observed entry (or faster if intermediate Hadamard products are cached). This keeps both memory and time independent of M. If q is moderate, one can also precompute and store all needed rows $z_t \equiv Z_{j_t,:}$ for $t = 1,\dots,q$ in an array of size $q\times r$, reducing the per-iteration cost from $O(qdr)$ to $O(qr)$ at the expense of $O(qr)$ memory.

Matvec formula

Given $X \in \mathbb{R}^{n\times r}$, compute $Y \in \mathbb{R}^{n\times r}$ so that $\operatorname{vec}(Y) = A\operatorname{vec}(X)$. Write $G \equiv KX \in \mathbb{R}^{n\times r}$. Then the observed predicted entries of $U \equiv KXZ^\top \in \mathbb{R}^{n\times M}$ are

$$u_t \equiv U_{i_t,j_t} = G_{i_t,:}\cdot Z_{j_t,:} \quad (t = 1,\dots,q), \tag{5}$$

each a length-r dot product. Now form the sparse matrix $\widetilde{U} \in \mathbb{R}^{n\times M}$ with $(\widetilde{U})_{i_t,j_t} = u_t$ and all other entries zero (this is exactly $\operatorname{reshape}(SS^\top \operatorname{vec}(U))$). Finally apply $(Z\otimes K)^\top$ using the transpose identity

$$(Z\otimes K)^\top \operatorname{vec}(\widetilde{U}) = \operatorname{vec}(K\,\widetilde{U}\,Z), \tag{6}$$

obtaining the main term $K(\widetilde{U}Z)$. Adding the Tikhonov term gives

$$Y = K\,(\widetilde{U}Z) + \lambda KX. \tag{7}$$

Proof of (7). Let $x = \mathrm{vec}(X)$. The first term satisfies $(Z \otimes K)^\top P (Z \otimes K) x = (Z \otimes K)^\top \mathrm{vec}(\widetilde{U})$ with $\widetilde{U} \equiv \mathrm{reshape}_{n \times M}\big(P\,\mathrm{vec}(KXZ^\top)\big)$. Using (6) yields $\mathrm{vec}(K\widetilde{U}Z)$, i.e., the matrix $K(\widetilde{U}Z)$. The regularizer contributes $\lambda(I_r \otimes K)\,\mathrm{vec}(X) = \lambda\,\mathrm{vec}(KX)$, giving (7).

How to compute $\widetilde{U}Z$ in $O(qr)$

We never form $\widetilde{U}$ explicitly as an $n \times M$ array. Instead, compute the product $H \equiv \widetilde{U}Z \in \mathbb{R}^{n\times r}$ by accumulating contributions from the q nonzeros:

$$H_{i_t,:} \mathrel{+}= u_t\, Z_{j_t,:} \qquad (t = 1, \ldots, q). \tag{8}$$

Each update is a SAXPY of length r, so the cost is $O(qr)$. Then compute KH in $O(n^2 r)$ time (dense K), and add λKX.

Implementation sketch (matvec). Given $X \in \mathbb{R}^{n\times r}$, precompute $G \leftarrow KX$. Initialize $H \leftarrow 0 \in \mathbb{R}^{n\times r}$. Loop over observed entries $t = 1, \ldots, q$: compute (or fetch) the Khatri–Rao row $z_t \equiv Z_{j_t,:}$, compute the scalar $u_t \leftarrow G_{i_t,:} z_t^\top$, and update $H_{i_t,:} \leftarrow H_{i_t,:} + u_t z_t$. Finally return $Y \leftarrow KH + \lambda G$.

In practice, the gather step can be batched as $u_t = \langle G_{i_t,:}, z_t\rangle$ via a single fused kernel (or vectorized BLAS), and the accumulation $H_{i_t,:} \mathrel{+}= u_t z_t$ can be implemented with a scatter-add primitive; both still cost $O(qr)$.

Matvec complexity. Assuming dense K:

- $G = KX$: $O(n^2 r)$.
- gather u_t via (5): $O(qr)$ given access to $Z_{j_t,:}$; if $Z_{j_t,:}$ is computed on the fly from the CP factors, add $O(q(d-1)r)$.
- accumulate $H = \widetilde{U}Z$ via (8): $O(qr)$ (plus the same cost to form $Z_{j_t,:}$ if needed).
- KH and λKX: $O(n^2 r)$ (can reuse G for KX).

Total per matvec: $O(n^2 r + qr)$ time if Z-rows are available (or $O(n^2 r + qdr)$ if computed on the fly), $O(nr + qr)$ memory for X, G, H and index lists; crucially independent of $N = nM$.

Computing the right-hand side b without forming T

The right-hand side can be written as $b = \mathcal{L}^\top y$ with $y = S^\top \mathrm{vec}(T)$. Scattering y back to an $n \times M$ matrix simply reconstructs T (zeros at missing entries), so by (6) one has $\mathcal{L}^\top y = \mathrm{vec}(K(TZ))$. Equivalently, $b = \mathrm{vec}(KB)$ with $B = TZ$. We can compute B using only observed entries: if the observed tensor value at (i_t, j_t) is t_t, then

$$B_{i_t,:} \mathrel{+}= t_t\, Z_{j_t,:},$$

which costs $O(qr)$ given access to $Z_{j_t,:}$ (or $O(qdr)$ if each $Z_{j_t,:}$ is formed on the fly from the CP factors), followed by KB in $O(n^2 r)$. (This is the same sparse accumulation pattern as (8).)

Optional change of variables (solve for $A_k = KW$)

Assume $K \succ 0$ and define the CP factor directly as $A \equiv A_k \equiv KW \in \mathbb{R}^{n\times r}$. Then (2) becomes

$$\min_{A\in\mathbb{R}^{n\times r}} \frac{1}{2}\|P\circ(T - AZ^\top)\|_F^2 + \frac{\lambda}{2}\,\mathrm{Tr}(A^\top K^{-1}A),$$

with normal equation

$$\Big[(Z\otimes I_n)^\top P(Z\otimes I_n) + \lambda(I_r\otimes K^{-1})\Big]\,\mathrm{vec}(A) = \mathrm{vec}(B), \qquad B = TZ.$$

This formulation replaces multiplications by K with solves in K; in particular, a matvec requires forming only observed entries of $AZ^\top$ (cost $O(qr)$) and applying K^{-1} to an $n\times r$ matrix (e.g. via a Cholesky factorization) at cost $O(n^2r)$. Concretely, for $X\in\mathbb{R}^{n\times r}$, the main term uses the same gather/scatter pattern as before but with $G\leftarrow X$ in (5) (no kernel multiply), and the regularizer adds $\lambda K^{-1}X$. A natural preconditioner in this variable is obtained by dropping the mask: $A_{0,A} = \alpha\big((Z^\top Z)\otimes I_n\big) + \lambda(I_r\otimes K^{-1})$ with $\alpha\approx q/N$. Since this is a Kronecker sum, it diagonalizes in the eigenbases of $Z^\top Z$ and K with eigenvalues $\alpha\sigma_a + \lambda/\lambda_b$. Equivalently, applying $A_{0,A}^{-1}$ to $\mathrm{vec}(R)$ amounts to solving $\alpha KXG + \lambda X = KR$ for X (a Sylvester-type equation). After solving for A, recover $W = K^{-1}A$. (Equivalence follows immediately by substituting $A = KW$ into (2); the corresponding normal equations match and the minimizers map bijectively when K is invertible.)

Preconditioning

Kronecker "full-observation" preconditioner

A standard and effective choice is to drop the mask P (equivalently, pretend all entries are observed), or to replace it by its mean under uniform sampling $P\approx\alpha I_N$ with $\alpha\equiv q/N$. This yields the Kronecker-structured approximation

$$A_0 \equiv \alpha(Z\otimes K)^\top(Z\otimes K) + \lambda(I_r\otimes K) = \alpha(Z^\top Z)\otimes(K^\top K) + \lambda(I_r\otimes K) = \alpha(Z^\top Z)\otimes(K^2) + \lambda(I_r\otimes K),$$

where K^2 denotes the usual matrix product KK (since K is symmetric). Let $G\equiv Z^\top Z\in\mathbb{R}^{r\times r}$. Since Z is a Khatri–Rao product, G can be computed without forming Z via the Hadamard product (denoted $*$) of Gram matrices:

$$G = \mathop{*}_{i\neq k}(A_i^\top A_i), \tag{9}$$

costing $O(\sum_{i\neq k} n_i r^2)$. Indeed, for columns $a,b\in[r]$, one has

$$\begin{aligned} G_{ab} &= \sum_{j\in[M]} Z_{j,a}Z_{j,b} \\ &= \sum_{i_1,\dots,i_{k-1},i_{k+1},\dots,i_d}\ \prod_{\ell\neq k} A_\ell(i_\ell,a)A_\ell(i_\ell,b) \\ &= \prod_{\ell\neq k}(A_\ell^\top A_\ell)_{ab}. \end{aligned}$$

If we precompute eigendecompositions

$$K = U\Lambda U^\top, \qquad G = V\Sigma V^\top,$$

then A_0 diagonalizes in the Kronecker basis:

$$A_0 = (V \otimes U)\,\mathrm{diag}\big(\alpha\sigma_a\lambda_b^2 + \lambda\lambda_b\big)_{a\in[r],b\in[n]}\,(V \otimes U)^\top.$$

Hence applying $M^{-1} \approx A_0^{-1}$ to a vector $x = \mathrm{vec}(X)$ can be done by:

1. transform $\widehat{X} \leftarrow U^\top X V$ (two small dense multiplies),
2. elementwise divide $\widehat{X}_{b,a} \leftarrow \widehat{X}_{b,a}/(\alpha\sigma_a\lambda_b^2 + \lambda\lambda_b)$,
3. inverse transform $X \leftarrow U\widehat{X}V^\top$.

Matrix-equation view (Sylvester form). For any $X \in \mathbb{R}^{n\times r}$,

$$A_0\,\mathrm{vec}(X) = \mathrm{vec}\big(\alpha K^2 X G + \lambda K X\big) = \mathrm{vec}\big(K(\alpha K X G + \lambda X)\big).$$

Thus applying A_0^{-1} to $\mathrm{vec}(R)$ is equivalent to solving the matrix equation

$$\alpha K X G + \lambda X = K^{-1} R. \tag{10}$$

Diagonalizing K and G gives the elementwise formula above; alternatively one may use Schur-based Sylvester solvers.

This costs $O(n^2 r + n r^2)$ per application (often dominated by $O(n^2 r)$ when $n \geq r$), after one-time setup $O(n^3 + r^3)$.

Why this is reasonable. If there are no missing entries ($P = I_N$) and we set $\alpha = 1$, then $A = A_0$ exactly. More generally, under uniform random sampling of entries one has $\mathbb{E}[P] = (q/N)I_N$, hence $\mathbb{E}[(Z\otimes K)^\top P(Z\otimes K)] = (q/N)(Z^\top Z \otimes K^2)$, so choosing $\alpha = q/N$ makes A_0 match the mean (and capture the dominant Kronecker spectral structure), while the deviation of P from αI acts as a perturbation that PCG corrects through iterations. Formally, write $C \equiv (Z \otimes K) \in \mathbb{R}^{N\times nr}$ and note that $C^\top P C$ is a sum of q sampled row outer products. Under uniform sampling, $\mathbb{E}[C^\top P C] = \alpha C^\top C$. Matrix Chernoff/concentration results imply that, under a mild incoherence/leverage-score condition on the rows of C, one has with high probability

$$(1-\varepsilon)\,\alpha\, C^\top C \;\preceq\; C^\top P C \;\preceq\; (1+\varepsilon)\,\alpha\, C^\top C, \tag{11}$$

provided $q \gtrsim \mu\,(nr)\log(nr)/\varepsilon^2$, where one possible choice is the row-coherence $\mu \equiv \frac{N}{\|C\|_F^2}\max_{s\in[N]}\|C_{s,:}\|_2^2$. Thus $C^\top P C$ is a small (sandwiched) perturbation of $\alpha C^\top C$ even though $\|P - \alpha I\|$ itself is not small.

Writing $A = A_0 + \Delta$ with $\Delta \equiv C^\top(P - \alpha I)C$, Eq. (11) yields $\|\Delta\| \leq \varepsilon\,\alpha\,\|C\|_2^2 = \varepsilon\,\alpha\,\|Z\|_2^2\,\|K\|_2^2$. Using Weyl's inequality gives

$$\lambda_i(A_0) - \|\Delta\| \leq \lambda_i(A) \leq \lambda_i(A_0) + \|\Delta\|.$$

Equivalently, for $M \equiv A_0$ the eigenvalues of $M^{-1}A = I + M^{-1}\Delta$ lie in $[1-\delta, 1+\delta]$ with

$$\delta \equiv \|M^{-1/2}\Delta M^{-1/2}\| \leq \frac{\|\Delta\|}{\lambda_{\min}(A_0)} \leq \varepsilon\,\frac{\alpha\,\|Z\|_2^2\,\|K\|_2}{\lambda}, \qquad (\text{since } \lambda_{\min}(A_0) \geq \lambda\,\lambda_{\min}(K)).$$

In particular, if $\delta < 1$ then $\kappa(M^{-1}A) \leq (1+\delta)/(1-\delta)$ and PCG converges in $O(\sqrt{\kappa}\,\log(1/\varepsilon))$ iterations. Note that $\|Z\|_2^2 = \lambda_{\max}(Z^\top Z) = \lambda_{\max}(G) = \|G\|_2$, and G is available via the Khatri–Rao Gram identity (9) (computed from the CP factor Gram matrices), so this estimate can be evaluated without ever forming Z.

Simpler (cheaper) preconditioners

If eigendecompositions are too costly, a cheaper alternative is a block-diagonal preconditioner

$$M_{\mathrm{bd}} = (\mathrm{diag}(G) \otimes K^2) + \lambda(I_r \otimes K),$$

which decouples the r components, requiring r solves with $n \times n$ matrices of the form $(g_{\ell\ell}K^2 + \lambda K) = K(g_{\ell\ell}K + \lambda I)$. If K is factored once (Cholesky), these are fast; if K is large, one can use an approximate factorization (pivoted Cholesky / incomplete Cholesky) as a preconditioner for these inner solves.

Overall complexity and scaling

Let m be the number of PCG iterations to reach a desired tolerance; standard theory gives $m = O\big(\sqrt{\kappa(M^{-1}A)}\,\log(1/\varepsilon)\big)$ for relative error ε, so a good preconditioner aims to make $\kappa(M^{-1}A)$ close to 1.

- One-time setup: compute Z and (optionally) $G = Z^\top Z$ in $O(Mr^2)$ if done naively, but in CP-ALS contexts Z is implicit and G is usually assembled from Hadamard products of Gram matrices of each factor at cost $O(\sum_{i \neq k} n_i r^2)$, avoiding M.
- Right-hand side: compute B in $O(qr)$ and then KB in $O(n^2 r)$.
- Each PCG iteration:
  - matvec Ax: $O(n^2 r + qr)$.
  - preconditioner apply (Kronecker-eig): $O(n^2 r + nr^2)$.
  - vector updates/inner products: $O(nr)$.

Hence total time $O\big(m(n^2 r + qr + nr^2) + qr + n^2 r\big)$, which is dramatically better than $O(n^3 r^3)$ when $m \ll n^2 r^2$ and $q \ll N$.

Key point: no N-scale work. All operations are expressed in terms of n, r, q and small Gram matrices; selection/scatter uses only the q observed indices and values.

Remark (faster kernel multiplies). If K admits a fast matrix–vector/matrix multiply (e.g., via Nyström, inducing points, random features, or a structured kernel), then the $O(n^2 r)$ terms above can be reduced accordingly; the $O(qr)$ sparse gather/accumulate terms remain unchanged.