Simultaneous matrix diagonalization algorithm for the tensor rank approximation problem

Rima Khouja

Élie Cartan de Lorraine (IECL) & Centre de Recherche en Automatique de Nancy (CRAN), France

Algebraic geometry and complexity theory workshop, AGATES, November 14th-18th, 2022, IM PAN, Poland

Contact: `rima.khouja@univ-lorraine.fr`

- **The Canonical Polyadic Decomposition** (CPD) or **rank decomposition** express the tensor as a minimal sum of rank-one simple tensors in the form

$$\mathcal{A} = \sum_{i=1}^{r} a_i^1 \otimes \ldots \otimes a_i^d, \tag{1}$$

where r is a positive integer and $a_i^k \in \mathbb{C}^{n_k}$, $\forall\ 1 \leq i \leq r$, $\forall\ 1 \leq k \leq d$.

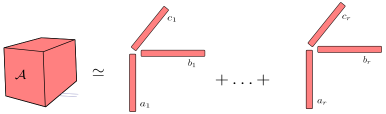

Figure 1: CPD of a three-dimensional tensor.

- The *factor matrices* are given by:

$$A_k = [a_1^k \ldots a_r^k] \in \mathbb{C}^{n_k \times r},\ \forall 1 \leq k \leq d.$$

- Using the factor matrices (1) can be expressed as follows

$$\mathcal{A} = [\![A_1, \ldots, A_d]\!] = \sum_{i=1}^{r} a_i^1 \otimes \ldots \otimes a_i^d.$$

- In practice measurements are rarely free of noise, consequently the rank decomposition of the tensor built from these measurements is rarely exact.
- **Low rank tensor approximation problem** aims to best approximate a given tensor $\mathcal{A} \in \mathbb{C}^{n_1 \times \dots \times n_d}$ by a rank decomposition with a number of components equal to the approximation rank r:

$$\min_{\hat{\mathcal{A}}} \|\mathcal{A} - \hat{\mathcal{A}}\|, \text{ with } \hat{\mathcal{A}} = \sum_{i=1}^{r} a_i^1 \otimes \dots \otimes a_i^d = [\![A_1, \dots, A_d]\!]. \tag{2}$$

- **Slices** of a tensor are obtained by fixing all the indices of the tensor except of two.

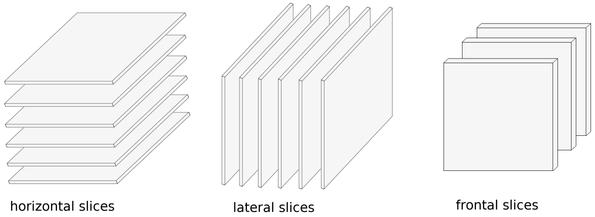

Figure 2: Slices of a third order tensor $\mathcal{A}[i,:,:]$, $\mathcal{A}[:,j,:]$, $\mathcal{A}[:,:,k]$[1].

- A pencil $M = [M_1, \ldots, M_s]$ of $n \times n$ real matrices corresponds to a tensor $\boldsymbol{M} \in \mathbb{R}^{n \times n \times s}$ where the slice $\boldsymbol{M}[:,:,i]$ is the matrix M_i.

[1]Credits to Felipe Bottega Diniz, thesis: *Tensor decompositions and algorithms, with applications to tensor learning, 2020.*

- The pencil M is simultaneously diagonalizable if there exists matrices $E = [E_1, \ldots, E_n] \in \mathrm{GL}(n)$, $F = [F_1, \ldots, F_n] \in \mathrm{GL}(n)$ with

$$M_k = F \operatorname{diag}(\Sigma_{[:,k]}) E^t = F \Sigma_{[k]} E^t \tag{3}$$

where $\Sigma = [\sigma_{i,j}] \in \mathbb{R}^{n \times s}$ and $\Sigma_{[:,k]}$ is the k^{th} column of Σ and $\Sigma_{[k]} = \operatorname{diag}(\Sigma_{[:,k]})$.

- In this case the corresponding tensor $\boldsymbol{M}$ can be written as

$$\boldsymbol{M} = \sum_{k=1}^{n} F_k \otimes E_k \otimes \Sigma_{[k,:]} \tag{4}$$

where F_k (resp. E_k) is the k^{th} column of F, (resp. E) and $\Sigma_{[k,:]} = [\sigma_{k,1}, \ldots, \sigma_{k,s}]$ is the k^{th} row of Σ.

- Conversely, if a tensor $\boldsymbol{M}$ can be written as (4), then the pencil $M = [\boldsymbol{M}[:,:,i]]$ is simultaneously diagonalizable and can be written as (3).

- In this case: Compute an approximate simultaneous matrix diagonalization of the pencil M is equivalent to compute an approximate rank-n decomposition to the associated tensor $\boldsymbol{M}$.

Plan

- Given a pencil of square matrices $M = [M_1, \ldots, M_s] \in \mathbb{R}^{n\times n}$. We address the following points:

  1. If the pencil of matrices is simultaneously diagonalizable:
     - We construct a Newton-type sequence that converges quadratically towards the solution $(E, F, (\Sigma_i)_{1\leq i\leq s})$.
     - We exhibit a certification test that the sequence converges towards the solution.
  2. If the pencil is not simultaneously diagonalizable:
     - The objective is to find two matrices E and F that diagonalize approximately the pencil, i.e. to compute an Approximate Simultaneous Diagonalization (ASD).
     - We present a Riemannian conjugate gradient algorithm to solve the ASD problem.
     - The algorithm is used to develop another algorithm that computes an approximated rank-r decomposition for tensors in $\mathbb{R}^{n_1\times n_2\times n_3}$ with r higher than the two first dimensional sizes, i.e. $r \geq \max(n_1, n_2)$.

Riemannian optimization

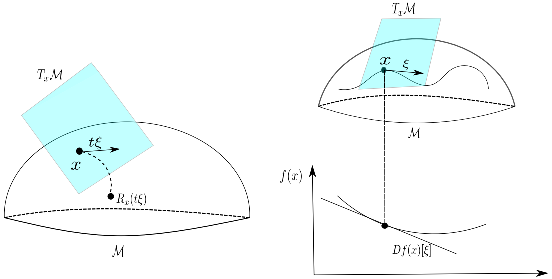

Figure 3: Illustration of one step of a typical Riemannian optimization method. On the right at x a descent direction ξ which leads to a decrease in f (i.e. $Df(x)[\xi] < 0$). On the left, the new point is defined on the manifold $\mathcal{M}$ by using a **retraction operator.**

Definition 0.1

A **retraction** R_p is a map $T_p\mathcal{M} \to \mathcal{M}$ that satisfies the following properties:

1. $R_p(0_p) = p$;
2. There exists an open neighborhood $\mathcal{U} \subset T_p\mathcal{M}$ of 0_p such that the restriction on $\mathcal{U}$ is well-defined and a smooth map;
3. Local rigidity$DR_p(0_p) = id_{T_p\mathcal{M}}$.

Riemannian conjugate gradient algorithm

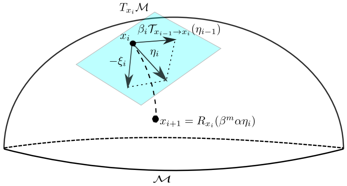
$T_{x_i}\mathcal{M}$
$\beta_i \mathcal{T}_{x_{i-1}\to x_i}(\eta_{i-1})$
x_i
η_i
$-\xi_i$
$x_{i+1} = R_{x_i}(\beta^m \alpha \eta_i)$
$\mathcal{M}$

Figure 4: Riemannian conjugate gradient algorithm with Armijo line-search at iteration i.

- $\xi_i := \operatorname{grad} f(x_i)$.
- $\eta_{i-1} \in T_{x_{i-1}}\mathcal{M}$ the conjugate direction at iteration $i-1$.
- $\mathcal{T}_{x_{i-1}\to x_i}$ a vector transport ☞ used to transport the tangent vector η_{i-1} in $T_{x_{i-1}}\mathcal{M}$ to a tangent vector in $T_{x_i}\mathcal{M}$.
- $\beta_i = \frac{\langle \operatorname{grad} f(x_i), \operatorname{grad} f(x_i) - \mathcal{T}_{x_{i-1}\to x_i}(\operatorname{grad} f(x_{i-1}))\rangle}{\langle \operatorname{grad} f(x_{i-1}), \operatorname{grad} f(x_{i-1})\rangle}$ (Polak-Ribière).
- Armijo backtracking step size ☞ find the smallest integer $m \geq 0$ such that for given $\alpha > 0, \beta, \sigma \in (0,1)$

$$f(x_i) - f(R_{x_i}(\beta^m \alpha \eta_i)) \geq -\sigma \langle \xi_i, \beta^m \alpha \eta_i \rangle.$$

➤ Newton-type method for simultaneous matrix diagonalization:

❍ Consider s simultaneously diagonalizable matrices $M_1, \cdots, M_s$ in $\mathbb{C}^{n\times n}$. We suppose that all the eigenvalues of M_i are simple. We aim to solve the following system of equations:

$$\begin{pmatrix} FE - I_n \\ FM_1E - \Sigma_1 \\ \vdots \\ FM_sE - \Sigma_s \end{pmatrix} = 0 \tag{5}$$

❍ Proposed strategies to solve (5):

1. Solve the system $(FE - I_n, FM_1E - \Sigma_1) = 0$.
2. Solve the system $(FM_1E - \Sigma_1, FM_2E - \Sigma_2) = 0$.
3. Take a linear combination M of $M_1, \ldots, M_s$, and solve $(FE - I_n, FME - \Sigma) = 0$.

☞ Deduce the diagonal matrices Σ_i by using the formula:

$$\Sigma_i = FM_iE.$$

❑ **Goal:** For each of these three systems → develop a **Newton-type sequence** that converge quadratically to the numerical solution + **certification** condition on the initial point.

❑ **Sketch from the resolution of system ❶ $(FE - I_n, FM_1E - \Sigma_1) = 0$:**

- We Consider the perturbations $E + EX$, $F + YF$, and $\Sigma + S$ respectively of E, F, and Σ, such that X and Y are in $\mathbb{C}^{n\times n}$ and S is a diagonal matrix in $\mathbb{C}^{n\times n}$. We get with $Z = FE - I_n$, and $\Delta = FME - \Sigma$:

$$\begin{aligned} &(F + YF)(E + EX) - I_n \\ &= Z + (Z + I_n)X + Y(Z + I_n) + Y(Z + I_n)X \end{aligned} \tag{6}$$

$$\begin{aligned} &(F + YF)M(E + EX) - \Sigma - S \\ &= FME - I_n + FMEX + YFME + YFMEX \\ &= \Delta - S + \Sigma X + Y\Sigma + \Delta X + Y\Delta + Y(\Delta + \Sigma)X \end{aligned} \tag{7}$$

- The Newton method consists in solving the linear system obtained from (6), (7):

$$\begin{cases} Z + X + Y & = 0 \\ \Delta - S + \Sigma X + Y\Sigma & = 0 \end{cases}$$

Lemma 0.2

Let $\Sigma = \mathrm{diag}(\sigma_1, \cdots, \sigma_n)$, $Z = (z_{i,j})_{1\le i,j\le n}$ *and* $\Delta = (\delta_{i,j})_{1\le i,j\le n}$ *be given matrices in* $\mathbb{C}^{n\times n}$. *Assume that* $\sigma_i \neq \sigma_j$ *for* $i \neq j$. *Let* S, X *and* Y *be matrices defined by*

$$\begin{aligned} S &= \mathrm{diag}(\Delta - Z\Sigma) && (8)\\ x_{i,i} &= 0 && (9)\\ x_{i,j} &= \frac{-\delta_{i,j} + z_{i,j}\sigma_j}{\sigma_i - \sigma_j}, \qquad i \neq j && (10)\\ y_{i,i} &= -z_{i,i} && (11)\\ y_{i,j} &= \frac{\delta_{i,j} - z_{i,j}\sigma_i}{\sigma_i - \sigma_j}, \qquad i \neq j. && (12) \end{aligned}$$

Then we have

$$\begin{aligned} Z + X + Y &= 0 && (13)\\ \Delta - S + \Sigma X + Y\Sigma &= 0. && (14) \end{aligned}$$

Theorem 0.3

Let $E_0, F_0 \in GL_n$ and $\Sigma_0 \in \mathcal{D}'_n$ be given such that they define the sequences for $i \geqslant 0$,

$$\begin{array}{rcl} Z_i &=& F_i E_i - I_n \\ \Delta_i &=& F_i M E_i - \Sigma_i \\ S_i &=& \mathrm{diag}(\Delta_i - Z_i \Sigma_i) \\ E_{i+1} &=& E_i(I_n + X_i) \\ F_{i+1} &=& (I_n + Y_i) F_i \\ \Sigma_{i+1} &=& \Sigma_i + S_i, \end{array}$$

where S_i, X_i and Y_i are defined by the formulas (8–12). Let us define $\kappa_0 = \max\left(1, \max_{i \neq j} \frac{1}{|\sigma_{0,i} - \sigma_{0,j}|}\right)$, $K_0 = \max(1, \max_i |\sigma_{0,i}|)$ and $\varepsilon_0 = \max(\kappa_0^2 K_0^2 \|Z_0\|, \kappa_0^2 K_0 \|\Delta_0\|)$. Assume that $\varepsilon_0 \leq 0.033$.
Then the sequences $(\Sigma_i, E_i, F_i)_{i \geqslant 0}$ converge quadratically to the solution of $(FE - I_n, FME - \Sigma) = 0$. More precisely E_0 and F_0 are invertible and
$\|E_i - E_\infty\| \leqslant 8.1 \times 2^{1-2^{i+1}} \|E_0\| \frac{\varepsilon_0}{\kappa K}$; $\|F_i - F_\infty\| \leqslant 8.1 \times 2^{1-2^{i+1}} \|F_0\| \frac{\varepsilon_0}{\kappa K}$.

Table 1: The computational results throughout 7 iterations for real diagonalizable matrix of size 10, order of perturbation $= 10^{-6}$, and precision 1024.

Iteration	$\varepsilon := \max(\kappa_0^2 K_0^2 \|Z_0\|, \kappa_0^2 K_0 \|\Delta_0\|) \leq 0.033$	err_{res}
1	0.00131	$9.33e{-6}$
2	$2.39e-8$	$1.06e{-10}$
3	$1.68e-18$	$7.49e{-21}$
4	$2.93e-38$	$1.31e{-40}$
5	$4.21e-78$	$1.87e{-80}$
6	$1.17e-157$	$5.24e{-160}$
7	$4.16e-288$	$6.20e{-293}$

✐ R. Khouja, B. Mourrain, J-C. Yakoubsohn. Newton-type methods for simultaneous matrix diagonalization. Calcolo journal, 2022.

➤ **Approximate simultaneous matrix diagonalization (ASD):**

❍ $f(E,F) = \frac{1}{2}\sum_k \|FM_kE^t - diag(FM_kE^t)\|^2 \mid (E,F) \in GL(n) \times GL(n)$.

❍ f must be **invariant to diagonal scaling** ☞ $f(E,F) = f(\Sigma_1 E, \Sigma_2 F)$, thus we redefine f as:
$f(E,F) = \frac{1}{2}\sum_k \|M_k - F^{-1}diag(FM_kE^t)E^{-t}\|^2 \mid (E,F) \in GL(n) \times GL(n)$.

❍ Since we can construct a sequence (Σ_i) of non-singular diagonal matrices converging toward a singular matrix, we have to add some contraints on the optimization set $GL(n) \times GL(n)$.

❑ We fix the norm of the rows of E and F using the oblique constraint:

$$diag(EE^t) = I_n, \; diag(FF^t) = I_n,$$

☞ $\mathcal{M}_n^o = \{Y \in GL(n) \mid diag(YY^t) = I_n\}$ is called the oblique manifold, it is a Riemannian submanifold of $GL(n)$.

❑ The Riemannian least square formulation associated to this approximation problem:
$\min_{(E,F)\in\mathcal{M}_n^o\times\mathcal{M}_n^o} f(E,F) := \frac{1}{2}\sum_k \|M_k - F^{-1}diag(FM_kE^t)E^{-t}\|^2$ (20).

❑ We introduce a **Riemannian Conjugate Gradient** algorithm (RCG) to solve (20):

```
Require: initial iterate $X_1 := (E_1, F_1) \in \mathcal{M}_n^o \times \mathcal{M}_n^o$, tangent vector $\eta_0 := (\eta^{E_0}, \eta^{F_0}) = (0, 0)$.
1: for $i = 1, 2, \ldots$ do
2:     Compute the gradient $\xi_i := \mathrm{grad}_{ob} f(X_i)$;
3:     Compute a conjugate direction $\eta_i := -\xi_i + \beta_i \mathcal{T}_{X_{i-1}\to X_i}(\eta_{i-1})$;
4:     Perform Armijo backtracking to find the smallest integer $m \geq 0$ such that
           $f(X_i) - f(R_{X_i}(0.5^m \eta_i)) \geq -0.1 \times 0.5^m \langle \xi_i, \eta_i \rangle^r_{X_i}$;
5:     Compute the next new point $X_{i+1} := R_{X_i}(0.5^m \eta_i)$;
6: end for
```

Algorithm 1: Riemannian conjugate gradient for ASD.

metric	$\langle \xi^B, \eta^B \rangle_B^{\ell} = \mathrm{tr}(B^{-1}\xi^B(B^{-1}\eta^B)^t)$	$\langle \xi^B, \eta^B \rangle_B^{r} = \mathrm{tr}(\xi^B B^{-1}(\eta^B B^{-1})^t)$
projection map onto $T_B\mathcal{M}_n^o$	$P_B^{ob,\ell}(Y) = Y - BB^t\Delta B$, Δ is a diagonal matrix with $\mathrm{diag}(\Delta) = (BB^t * BB^t)^{-1}\,\mathrm{diag}(YB^t)$	$P_B^{ob,r}(Y) = Y - \Delta BB^tB$, $\Delta = \mathrm{ddiag}(YB^t)\,\mathrm{ddiag}((BB^t)^2)^{-1}$
gradient	$\mathrm{grad}_{ob}^{\ell} f(B) = P_B^{ob,\ell}(\mathrm{grad}_{\ell} f(B))$	$\mathrm{grad}_{ob}^{r} f(B) = P_B^{ob,r}(\mathrm{grad}_{r} f(B))$
retraction	$R_B^{ob,\ell}(\xi^B) = \Delta(\exp_B^{\ell}(\xi^B))\exp_B^{\ell}(\xi^B)$ with $\Delta(Y) = \mathrm{ddiag}(YY^t)^{-\frac{1}{2}}$	$R_B^{ob,r}(\xi^B) = \Delta(\exp_B^{r}(\xi^B))\exp_B^{r}(\xi^B)$
vector transport	$\mathcal{T}^{ob,\ell}_{B\to R_B^{ob,\ell}(\xi^B)}(\eta^B) = P^{ob,\ell}_{R_B^{ob,\ell}(\xi^B)}(R_B^{ob,\ell}(\xi^B)B^{-1}\eta^B)$	$\mathcal{T}^{ob,r}_{B\to R_B^{ob,r}(\xi^B)}(\eta^B) = P^{ob,r}_{R_B^{ob,r}(\xi^B)}(\eta^B B^{-1}R_B^{ob,r}(\xi^B))$

Table 2: Main tools for Riemannian optimization on the oblique manifold $\mathcal{M}_n^o$ [2].

Definition 0.4

For $(E,F) \in \mathcal{M}_n^o \times \mathcal{M}_n^o$, $\xi = (\xi^E, \xi^F)$, $\eta = (\eta^E, \eta^F) \in T_E\mathcal{M}_n^o \times T_F\mathcal{M}_n^o \simeq T_{(E,F)}\mathcal{M}_n^o \times \mathcal{M}_n^o$, then $\langle \xi, \eta \rangle_{(E,F)}^r = \langle (\xi^E, \xi^F), (\eta^E, \eta^F) \rangle_{(E,F)}^r = \langle \xi^E, \eta^E \rangle_E^r + \langle \xi^F, \eta^F \rangle_F^r$.

[2] F. Bouchard, B. Afsari, J. Malick, and M. Congedo. Approximate joint diagonalization with Riemannian optimization on the general linear group. SIMAX, 2020

Computation of the gradient

- Let $f(E,F) = \sum_{k=1}^{s} f_k(E,F)$ such that,

$$f_k(E,F) = \frac{1}{2}\|M_k - F^{-1}\,\mathrm{ddiag}(FM_kE^t)E^{-t}\|^2.$$

- Let $f_k^F := f_k(.,F) : \mathcal{M}_n^o \to \mathbb{R},\ E \mapsto f_k(E,F)$;
  $f_k^E := f_k(E,.) : \mathcal{M}_n^o \to \mathbb{R},\ F \mapsto f_k(E,F)$.

- We have,
  $$\mathrm{grad}_{ob} f(E,F) = \left(\sum_{k=1}^{s} \mathrm{grad}_{ob} f_k^F(E), \sum_{k=1}^{s} \mathrm{grad}_{ob} f_k^E(F) \right).$$

- Let us consider $\mathrm{grad}_{ob}\, f_k^F(E)$ and the same procedure can be applied to compute $\mathrm{grad}_{ob}\, f_k^E(F)$.

- We have $\mathrm{grad}_{ob}\, f_k^F(E) = P_{ob,r}^E(\mathrm{grad}\, f_k^F(E))$, where $\mathrm{grad}\, f_k^F(E)$ is the Riemannian gradient of f_k^F at $E \in \mathrm{GL}_n$ equipped with the right-invariant metric given by

$$\mathrm{grad}\, f_k^F(E) = \mathrm{grad}_{\mathrm{Euc}}\, f_k^F(E) E^t E,$$

  such that $\mathrm{grad}_{\mathrm{Euc}}\, f_k^F(E)$ is the Euclidean gradient of f_k^F at $E \in \mathrm{GL}_n$.
- Finally, to compute $\mathrm{grad}_{\mathrm{Euc}}\, f_k^F(E)$ we compute the time derivative of f_k^F $(\dot{f}_k^F)$ in terms of the time derivative of E $(\dot{E} \in T_E\mathrm{GL}_n)$ by using the trace identities in order to obtain

$$\dot{f}_k^F(E) = \mathrm{tr}(X\dot{E}),$$

  on the other hand we have that $\dot{f}_k^F(E) = \mathrm{tr}((\mathrm{grad}_{\mathrm{Euc}}\, f_k^F(E))^t \dot{E})$, herein we can deduce $\mathrm{grad}_{\mathrm{Euc}}\, f_k^F(E) = X^t$.
- The formulas of $\mathrm{grad}_{\mathrm{Euc}}\, f_k^F(E)$ and $\mathrm{grad}_{\mathrm{Euc}}\, f_k^E(F)$ are given by:
  $\mathrm{grad}_{\mathrm{Euc}}\, f_k^F(E) = [Q_k(E,F)\,\mathrm{ddiag}(\mathrm{FM}_k\, E^t) - \mathrm{ddiag}(Q_k(E,F))\,\mathrm{FM}_k\, E^t]E^{-t}$,
  $\mathrm{grad}_{\mathrm{Euc}}\, f_k^E(F) = [Q_k(E,F)^t\,\mathrm{ddiag}(\mathrm{FM}_k\, E^t) - \mathrm{ddiag}(Q_k(E,F))\,\mathrm{EM}_k^t\, F^t]F^{-t}$,
  with $Q_k(E,F) = (EE^t)^{-1}(\mathrm{EM}_k^t\, F^t - \mathrm{ddiag}(\mathrm{FM}_k\, E^t))(FF^t)^{-1}$.

- Some points can be handled in this algorithm:
  - Enhance the stopping criterion.
  - Some analysis for the convergence properties.
  - Compute the computational complexity per iteration and comparison in this term with the standard algorithms to solve the rank-n tensor approximation of a tensor in $\mathbb{R}^{n\times n\times s}$.

➤ **ASD for tensor rank approximation problem:**

❑ For a tensor $\boldsymbol{M} \in \mathbb{R}^{n\times n\times s}$ compute an approximate simultaneous matrix diagonalization for the associate pencil of matrices M ☞ compute an approximate rank-n tensor decomposition.

❑ Suppose that we have a tensor $\boldsymbol{M} \in \mathbb{R}^{n_1\times n_2\times n_3}$ that we aim to approximate to a tensor of rank $r \geq \max(n_1, n_2)$, i.e. find three factor matrices $A,\ B,\ C$ respectively in $\mathbb{R}^{n_1\times r}$, $\mathbb{R}^{n_2\times r}$ and $\mathbb{R}^{n_3\times r}$, that minimize the following non-linear least-squares function:

$$f(A,B,C) = \tfrac{1}{2}\Big\|\boldsymbol{M} - \sum_{i=1}^{r} A[:,i]\otimes B[:,i]\otimes C[:,i]\Big\|^2.$$

- We present an alternate optimization method, to find such factor matrices A, B and C, such that at each iteration it alternates between (RCG) and the resolution of the least-squares problem.

❑ The algorithm can be summarized as follows:

- Let M be the pencil of n_3 matrices of size $n_1 \times n_2$ associated to $\boldsymbol{M}$.
- Let $\tilde{M} = [\tilde{M}_1, \dots, \tilde{M}_{n_3}]$ with $\tilde{M}_i$ of size $r \times r$, such that $\tilde{M}_i[1:n_1, 1:n_2] = M_i, \ \forall\ 1 \leq i \leq n_3$.
- Approximate the pencil $\tilde{M}$ locally by a pencil of simultaneously diagonalizable matrices $\tilde{M}'$ by using (RCG), i.e. use (RCG) to find $(E, F) \in \mathcal{M}_r^o \times \mathcal{M}_r^o$ that solves locally:

$$\min_{(E,F)\in\mathcal{M}_r^o\times\mathcal{M}_r^o} \frac{1}{2}\sum_{k=1}^{n_3} \|\tilde{M}_k - F^{-1}\,\mathrm{ddiag}(F\tilde{M}_k E^t)E^{-t}\|^2.$$

- The tensor associated to the pencil $\tilde{M}'$ can be written as a tensor rank-r decomposition with the factors E, F and Σ. This means, that the tensor associated to $\tilde{M}$ is approximated locally by the tensor rank-r decomposition given by $\tilde{M}'$.

- The distance function can also be reduced, by fixing this time the obtained matrices E, F and optimizing all the entries of the matrices $\tilde{M}_k$ except of those in the fixed blocks:

$$\begin{aligned} &\min_{\tilde{M}} \frac{1}{2} \sum_{k=1}^{n_3} \| \tilde{M}_k - F^{-1} \operatorname{ddiag}(F \tilde{M}_k E^t) E^{-t} \|^2, \\ &\text{s.t. } \tilde{M}_k[1:n_1, 1:n_2] = M_k, \text{ for } k \in \{1, \dots, n_3\}. \end{aligned} \tag{15}$$

- In summary, we minimize the function

$$\begin{aligned} f(E, F, \tilde{M}) &= \frac{1}{2} \sum_{k=1}^{n_3} \| \tilde{M}_k - F^{-1} \operatorname{ddiag}(F \tilde{M}_k E^t) E^{-t} \|^2 \\ \text{s.t. } &(E, F) \in \mathcal{M}_r^o \times \mathcal{M}_r^o, \\ &\tilde{M}_k[1:n_1, 1:n_2] = M_k, \text{ for } k \in \{1, \dots, n_3\}, \end{aligned} \tag{16}$$

by *alternating* between two optimization minimization problems.

- At the end, an approximated rank-r decomposition is obtained for the tensor associated to a pencil $\tilde{M}$ from which we can extract an approximated rank-r decomposition for the targeted tensor $\boldsymbol{M}$.

```
Require: $\boldsymbol{M} \in \mathbb{R}^{n_1 \times n_2 \times n_3}$, approximation rank $r \geq \max(n_1, n_2)$.
1: Compute $M$ the pencil associated to $\boldsymbol{M}$;
2: Extend $M$ to the pencil $\tilde{M}$ of $n_3$ matrices, such that $\tilde{M}_k[1:n_1, 1:n_2] = M_k$
   for $k \in \{1, \dots, n_3\}$; Set initial iterate $(E_0, F_0, \tilde{M}_0 := \tilde{M})$, with
   $(E_0, F_0) \in \mathcal{M}_r^o \times \mathcal{M}_r^o$.
3: for $i = 1, 2, \dots$ do
4:     Obtain $(E_i, F_i)$ by applying RCG initialized by $(E_{i-1}, F_{i-1})$ to solve:
       $\min_{(E,F)\in\mathcal{M}_r^o\times\mathcal{M}_r^o} f_1(E,F) = \min_{(E,F)\in\mathcal{M}_r^o\times\mathcal{M}_r^o} \frac{1}{2}\sum_{k=1}^{n_3} \|\tilde{M}_{i-1,k} - F^{-1}\,\mathrm{ddiag}(F\tilde{M}_{i-1,k}E^t)E^{-t}\|^2.$
5:     Fix $(E_i, F_i)$ and update $\tilde{M}_{i-1}$ to $\tilde{M}_i$ by solving:
       $\min_{\tilde{M}} f_2(\tilde{M}) = \min_{\tilde{M}} \frac{1}{2}\sum_{k=1}^{n_3} \|\tilde{M}_k - F_i^{-1}\,\mathrm{ddiag}(F_i\tilde{M}_k E_i^t)E_i^{-t}\|^2,$   (17)
       s.t. $\tilde{M}_k[1:n_1, 1:n_2] = M_k,$ for $k \in \{1, \dots, n_3\}$.
6: end for
7: Extract factor matrices $A \in \mathbb{R}^{n_1 \times r}$, $B \in \mathbb{R}^{n_2 \times r}$, $C \in \mathbb{R}^{n_3 \times r}$, such that:
   $A = F_{\mathrm{end}}^{-1}[1:n_1, 1:r]$;
   $B = E_{\mathrm{end}}^{-t}[1:n_2, 1:r]$;
   $C = \Sigma^t$, where $\Sigma$ is of size $r \times n_3$, such that $\Sigma[:,k] = \mathrm{diag}(F_{\mathrm{end}}\tilde{M}_{\mathrm{end},k}E_{\mathrm{end}}^t)$.
```

Algorithm 2: Alternate optimization algorithm (AO) for tensor rank approximation of tensors in $\mathbb{R}^{n_1 \times n_2 \times n_3}$ with $r \geq \max(n_1, n_2)$.

Solving the least-squares problem

- For each k in $\{1, \dots, n_3\}$, we consider the linear least-squares problem

$$\min_{\tilde{M}_k \in \mathbb{R}^{r \times r}} \frac{1}{2} \| \tilde{M}_k - F^{-1} \operatorname{ddiag}(F \tilde{M}_k E^t) E^{-t} \|^2, \text{ s.t. } \tilde{M}_k[1 : n_1, 1 : n_2] = M_k.$$

- Let $H := \tilde{M}_k - F^{-1} \operatorname{ddiag}(F \tilde{M}_k E^t) E^{-t} \in \mathbb{R}^{r \times r}$. We have $h_{i,j} = \langle e_{i,j}, H \rangle$, where $(e_{i,j})_{1 \leq i,j \leq r}$ is the canonical basis of $\mathbb{R}^{r \times r}$.
- By developing the calculus using the trace properties in order to isolate our targeted variable $\tilde{M}_k$, we find:

$$h_{i,j} = \langle e_{i,j} - F^t \operatorname{ddiag}(E^{-1}[j, :] \otimes F^{-1}[i, :]) E, \tilde{M}_k \rangle.$$

- Hence, minimizing the Frobenius norm of $h_{i,j}$ is equivalent to minimizing the Frobenius norm of $(\operatorname{vec}(e_{i,j} - F^t \operatorname{ddiag}(E^{-1}[j, :] \otimes F^{-1}[i, :]) E))^t \operatorname{vec}(\tilde{M}_k)$. This leads us to the following system:

$$GX = 0,$$

where $G = [(\operatorname{vec}(e_{i,j} - F^t \operatorname{ddiag}(E^{-1}[j, :] \otimes F^{-1}[i, :]) E))^t] \in \mathbb{R}^{r^2 \times r^2}$,
$X = \operatorname{vec}(\tilde{M}_k) \in \mathbb{R}^{r^2}$.

- Recall that we have $n_1 n_2$ fixed entries in $\tilde{M}_k$ equal to the $n_1 n_2$ entries of M_k, thus we write the system $GX = 0$ as

$$G = [G_1, G_2],$$

$$X = [X_1, X_2],$$

  such that:
  - $X_2 \in \mathbb{R}^{n_1 n_2}$ represents the fixed known entries in $\tilde{M}_k$,
  - $G_2 \in \mathbb{R}^{r^2 \times n_1 n_2}$ contains the corresponding columns to X_2 in G.
- Hence, the system to solve becomes:

$$G_1 X_1 = Y, \tag{18}$$

  where $Y := -G_2 X_2$. Consequently, we can write

$$X_1 = G_1^{\dagger} X_2.$$

Extract the approximate rank decomposition

- The three factors A, B and C are computed by simply taking the block

$$
\begin{aligned}
&(\tilde{M}_k - F_{\text{end}}^{-1}\,\text{ddiag}(F_{\text{end}}\tilde{M}_k E_{\text{end}}^t)E_{\text{end}}^{-t})[1:n_1, 1:n_2] \\
=&M_k - (F_{\text{end}}^{-1}\,\text{ddiag}(F_{\text{end}}\tilde{M}_k E_{\text{end}}^t)E_{\text{end}}^{-t})[1:n_1, 1:n_2],
\end{aligned}
$$

  where

$$
\begin{aligned}
&(F_{\text{end}}^{-1}\,\text{ddiag}(F_{\text{end}}\tilde{M}_k E_{\text{end}}^t)E_{\text{end}}^{-t})[1:n_1, 1:n_2] \\
=&F_{\text{end}}^{-1}[1:n_1, 1:r]\,\text{ddiag}(F_{\text{end}}\tilde{M}_k E_{\text{end}}^t)E_{\text{end}}^{-t}[1:r, 1:n_2].
\end{aligned}
$$

- Thus the pencil M is approximated by the pencil $[F_{\text{end}}^{-1}[1:n_1, 1:r]\,\text{ddiag}(F_{\text{end}}\tilde{M}_k E_{\text{end}}^t)E_{\text{end}}^{-t}[1:r, 1:n_2]]_{1\leq k\leq n_3}$ which correspond to a rank-r decomposition with A, B and C given by:

$$
\begin{aligned}
A &= F_{\text{end}}^{-1}[1:n_1, 1:r]; \\
B &= E_{\text{end}}^{-t}[1:n_2, 1:r]; \\
C &= \Sigma^t \in \mathbb{R}^{n_3\times r},\ \text{where } \Sigma[:,k] = \text{diag}(F_{\text{end}}\tilde{M}_{\text{end},k}E_{\text{end}}^t).
\end{aligned}
$$

Example

- Let $T \in \mathbb{R}^{4\times 4\times 4}$ be the tensor associated to the matrix multiplication of square matrices of size 2×2:
$$T[1,:,:] = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix},\ T[2,:,:] = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix},\ T[3,:,:] = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \end{bmatrix},\ T[4,:,:] = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}.$$
- T is of rank 7. We apply the alternate optimization algorithm to find a rank-7 decomposition of T.
- Extend the pencil of the four matrices of size 4×4 to a pencil of matrices of size 7×7.
- Starting with a random initial point that gives an initial error

$$\mathrm{err}_0 := \frac{1}{2}\Big\| T - \sum_{i=1}^{7} A_0[:,i] \otimes B_0[:,i] \otimes C_0[:,i] \Big\|^2 = 412.6.$$

$$\xRightarrow{final\ \ error}$$

$$\mathrm{err}_{\mathrm{end}} := \tfrac{1}{2}\big\| T - \textstyle\sum_{i=1}^{7} A_{\mathrm{end}}[:,i] \otimes B_{\mathrm{end}}[:,i] \otimes C_{\mathrm{end}}[:,i] \big\|^2 = 3.19e - 12.$$

- The three factor matrices $A,\ B,\ C \in \mathbb{R}^{4\times 7}$ given by the algorithm are:

$$A = \begin{pmatrix} -3.29804 & -4.1045 & 3.19274 & -0.867293 & 0.22113 & 0.0994379 & -1.09832 \\ -2.23989 & -1.90246 & 0.718265 & -0.195113 & -0.408636 & -0.183755 & -0.745938 \\ 0.364624 & 1.03971 & -0.352981 & 1.5467 & -0.394356 & 0.112363 & -1.24109 \\ 0.247625 & -0.872426 & -0.0794047 & 0.347959 & 0.728746 & -0.207641 & -0.842896 \end{pmatrix},$$

$$B = \begin{pmatrix} -0.0807727 & -1.12551 & 1.16343 & 0.0966848 & -0.774754 & -0.846987 & 0.00372952 \\ 2.56926 & -0.723095 & -1.77348 & -0.147383 & -0.808044 & -0.883381 & -0.118618 \\ 0.0715004 & -0.74756 & 0.65238 & 0.874531 & 0.685632 & -0.474938 & 0.0337356 \\ -2.27374 & 3.29779 & -0.994464 & -1.3331 & 0.715093 & -0.495345 & -1.07291 \end{pmatrix},$$

$$C = \begin{pmatrix} -0.0537698 & 0.0635183 & 0.174347 & 0.737993 & -0.237901 & -1.15977 & 0.800897 \\ 0.239012 & 0.270613 & -0.256709 & 0.39936 & 1.05748 & 1.70766 & 0.4334 \\ 0.0515551 & 0.118177 & -0.167164 & 0.0232047 & -0.00748016 & -0.760829 & 0.5254 \\ -0.229165 & -0.162557 & 0.246134 & 0.012557 & 0.0332504 & 1.12025 & 0.284317 \end{pmatrix}.$$

How this approach can be important?

- Three-dimensional tensors appear in many applications for example in Signal Processing [A. Cichocki et al, 2015], in Chemometrics [R. Bro, 1997].
- In practice, the scenario of rank exceeds the mode size can quite occurs, for example in Blind Source Identification problem the number of sources can exceeds the number of sensors yielding an underdetermined system.
- In this case, it is often observed that applying a general tensor rank approximation method (broadly divided to ALS algorithms and quasi-Newton algorithms) could have some numerical difficulties.
- Herein extend the tensor as was described to skip the under-determination issue and applying the alternate optimization algorithm which basically relies on the resolution of two simple problems: the simultaneous diagonalization problem and the linear least-squares problem, could have an efficient numerical performance in this context.

Final comments

$\xrightarrow{\text{Perspectives}}$ ...

- Compute the computational complexity of the algorithm (the rank should exceeds the mode size moderately).
- The numerical stability of the algorithm: In general inherits from alternate optimization method the sensibility near ill-conditionning points and becomes slow ☞ proposing a strategy to avoid ill-conditioned points.

Thank you!