Adaptive Algebraic Multigrid for Canonical Tensor Decomposition

Hans De Sterck and Killian Miller
Department of Applied Mathematics
University of Waterloo, Canada

Weizmann Workshop 2013 on Multilevel Computational Methods and Optimization
(1) canonical tensor decomposition

- tensor = element of tensor product of real vector spaces ($N$-dimensional array)
- $N=3$:

  ![Diagram](image.png)

  (from “Tensor Decompositions and Applications”, Kolda and Bader, SIAM Rev., 2009 [1])

- canonical decomposition: decompose tensor in sum of $R$ rank-one terms (approximately)
canonical tensor decomposition

OPTIMIZATION PROBLEM

given tensor $\mathcal{T} \in \mathbb{R}^{I_1 \times \cdots \times I_N}$, find rank-$R$
canonical tensor $\mathcal{A}_R \in \mathbb{R}^{I_1 \times \cdots \times I_N}$ that minimizes

$$f(\mathcal{A}_R) = \frac{1}{2} \| \mathcal{T} - \mathcal{A}_R \|_F^2.$$ 

FIRST-ORDER OPTIMALITY EQUATIONS

$$\nabla f(\mathcal{A}_R) = g(\mathcal{A}_R) = 0.$$ 

(problem is non-convex, multiple (local) minima, solution may not exist
(ill-posed), ... ; but smooth, and we assume there is a local minimum)

(de Silva and Lim, SIMAX, 2009)
link with singular value decomposition

• SVD of \( A \in \mathbb{R}^{m \times n} \quad m \geq n \)

\[
A = U \Sigma V^t = \sigma_1 u_1 v_1^T + \ldots + \sigma_n u_n v_n^T
\]

• canonical decomposition of tensor
a difference with the SVD

truncated SVD is best rank-$R$ approximation:

$$A = \sigma_1 u_1 v_1^T + \ldots + \sigma_R u_R v_R^T + \sigma_{R+1} u_{R+1} v_{R+1}^T + \ldots + \sigma_n u_n v_n^T$$

$$\text{arg min} \quad \|A - B\|_F = \sigma_1 u_1 v_1^T + \ldots + \sigma_R u_R v_R^T$$

BUT best rank-$R$ tensor cannot be obtained by truncation: different optimization problems for different $R$!

given tensor $\mathcal{T} \in \mathbb{R}^{I_1 \times \ldots \times I_N}$, find rank-$R$ canonical tensor $\mathcal{A}_R \in \mathbb{R}^{I_1 \times \ldots \times I_N}$ that minimizes

$$f(\mathcal{A}_R) = \frac{1}{2} \|\mathcal{T} - \mathcal{A}_R\|_F^2.$$
Tensor approximation applications

(1) “Discussion Tracking in Enron Email Using PARAFAC” by Bader, Berry and Browne (2008) (sparse, nonnegative)
(2) chemometrics: analyze spectrofluorometer data (dense) (Bro et al., http://www.models.life.ku.dk/nwaydata1)
- 5 x 201 x 61 tensor: 5 samples (with different mixtures of three amino acids), 61 excitation wavelengths, 201 emission wavelengths
- goal: recover emission spectra of the three amino acids (to determine what was in each sample, and in which concentration)
(2) chemometrics: analyze spectrofluorometer data (dense) (Bro et al., http://www.models.life.ku.dk/nwaydata1)

- $5 \times 201 \times 61$ tensor: 5 samples (with different mixtures of three amino acids), 61 excitation wavelengths, 201 emission wavelengths
- goal: recover emission spectra of the three amino acids (to determine what was in each sample, and in which concentration)
- also: psychometrics, ..
tensor approximation applications

(3) “All-at-once Optimization for Coupled Matrix and Tensor Factorizations” by Acar, Kolda and Dunlavy (2011)

\[
f(A, B, C, V) = \| X - [A, B, C] \|^2 + \| Y - AV^T \|^2
\]

\[
\| W * (X - [A^{(1)}, ..., A^{(N)}]) \|^2 + \frac{1}{2} \| Y - A^{(n)} V^T \|^2
\]
(2) ‘workhorse’ algorithm: alternating least squares (ALS)

\[ f(\mathcal{A}_R) = \frac{1}{2} \left\| \mathcal{T} - \sum_{r=1}^{R} a_r^{(1)} \circ a_r^{(2)} \circ a_r^{(3)} \right\|_F^2 \]

(1) freeze all \( a_r^{(2)}, a_r^{(3)} \), compute optimal \( a_r^{(1)} \) via a least-squares solution (linear, overdetermined)
(2) freeze \( a_r^{(1)}, a_r^{(3)} \), compute \( a_r^{(2)} \)
(3) freeze \( a_r^{(1)}, a_r^{(2)} \), compute \( a_r^{(3)} \)

• repeat
alternating least squares (ALS)

\[ f(\mathbf{A}_R) = \frac{1}{2} \left\| \mathbf{T} - \sum_{r=1}^{R} a_r^{(1)} \circ a_r^{(2)} \circ a_r^{(3)} \right\|_F^2 \]

- “simple iterative optimization method”
- ALS is block nonlinear Gauss-Seidel
- ALS is monotone
- ALS is sometimes fast, but can also be extremely slow (depending on problem and initial condition)
alternating least squares (ALS)

\[
f(A_R) = \frac{1}{2} \left\| T - \sum_{r=1}^{R} a_r^{(1)} \circ a_r^{(2)} \circ a_r^{(3)} \right\|_F^2
\]

fast case

slow case

(we used Matlab with Tensor Toolbox (Bader and Kolda) and Poblano Toolbox (Dunlavy et al.) for all computations)
convergence acceleration for ALS (nonlinear optimization)

<table>
<thead>
<tr>
<th>convergence acceleration for linear systems</th>
<th>convergence acceleration for nonlinear optimization</th>
</tr>
</thead>
<tbody>
<tr>
<td>P-CG</td>
<td>NCG, P?</td>
</tr>
<tr>
<td>P-GMRES</td>
<td>?</td>
</tr>
<tr>
<td>MG</td>
<td>? (MG/OPT)</td>
</tr>
</tbody>
</table>
convergence acceleration for ALS (nonlinear optimization)

<table>
<thead>
<tr>
<th>convergence acceleration for linear systems</th>
<th>convergence acceleration for nonlinear optimization</th>
<th>convergence acceleration for nonlinear systems</th>
</tr>
</thead>
<tbody>
<tr>
<td>P-CG</td>
<td>NCG, P?</td>
<td>NCG, P?</td>
</tr>
<tr>
<td>P-GMRES</td>
<td>?</td>
<td>P-NGMRES (Washio and Oosterlee, Anderson)</td>
</tr>
<tr>
<td>MG</td>
<td>? (MG/OPT)</td>
<td>FAS</td>
</tr>
</tbody>
</table>

[Graphs showing convergence plots]
convergence acceleration for ALS (nonlinear optimization)

<table>
<thead>
<tr>
<th>convergence acceleration for linear systems</th>
<th>convergence acceleration for nonlinear optimization</th>
<th>convergence acceleration for nonlinear systems</th>
<th>convergence acceleration for nonlinear optimization</th>
</tr>
</thead>
<tbody>
<tr>
<td>P-CG</td>
<td>NCG, P?</td>
<td>NCG, P?</td>
<td>P-NCG</td>
</tr>
<tr>
<td>P-GMRES</td>
<td>?</td>
<td>P-NGMRES (Washio and Oosterlee, Anderson)</td>
<td>P-NGMRES for optimization</td>
</tr>
<tr>
<td>MG</td>
<td>?</td>
<td>FAS</td>
<td>adaptive AMG-FAS for optimization</td>
</tr>
</tbody>
</table>

![Graphs showing convergence acceleration for different methods](image-url)
Algorithm 1: N-GMRES optimization algorithm (window size \( w \))

**Input:** \( w \) initial iterates \( u_0, \ldots, u_{w-1} \).

\( i = w - 1 \)

repeat

**STEP I:** (generate preliminary iterate by one-step update process \( M(,.) \))
\[ \bar{u}_{i+1} = M(u_i) \]

**STEP II:** (generate accelerated iterate by nonlinear GMRES step)
\[ \hat{u}_{i+1} = \text{gmrres}(u_{i-w+1}, \ldots, u_i; \bar{u}_{i+1}) \]

**STEP III:** (generate new iterate by line search process)
\[ u_{i+1} = \text{linesearch}(\bar{u}_{i+1} + \beta(\hat{u}_{i+1} - \bar{u}_{i+1})) \]

\( i = i + 1 \)

until convergence criterion satisfied
step II: N-GMRES acceleration: $\nabla f(A_R) = g(A_R) = 0$

$$\hat{u}_{i+1} = \bar{u}_{i+1} + \sum_{j=0}^{i} \alpha_j (\bar{u}_{i+1} - u_j)$$

$$g(\hat{u}_{i+1}) \approx g(\bar{u}_{i+1}) + \sum_{j=0}^{i} \left. \frac{\partial g}{\partial u} \right|_{\bar{u}_{i+1}} \alpha_j (\bar{u}_{i+1} - u_j)$$

$$\approx g(\bar{u}_{i+1}) + \sum_{j=0}^{i} \alpha_j (g(\bar{u}_{i+1}) - g(u_j))$$

find coefficients $(\alpha_0, \ldots, \alpha_i)$ that minimize

$$\|g(\bar{u}_{i+1}) + \sum_{j=0}^{i} \alpha_j (g(\bar{u}_{i+1}) - g(u_j))\|_2.$$

UNIVERSITY OF WATERLOO
numerical results for ALS-preconditioned N-GMRES applied to tensor problem

- dense test problem (from Tomasi and Bro; Acar et al.):
  - random rank-$R$ tensor modified to obtain specific column collinearity, with added noise

\[ h(\mathcal{A}_R^{(i)}) = \frac{\| \mathcal{T} - \mathcal{A}_R^{(i)} \|_F}{\| \mathcal{T} \|_F} \]
nonlinearly preconditioned N-GMRES for nonlinear optimization

- works well for canonical tensor decomposition (N-GMRES accelerates ALS)
- also promising as a general way to accelerate ‘simple’ optimization methods

convergence acceleration for ALS (nonlinear optimization)

<table>
<thead>
<tr>
<th>convergence acceleration for linear systems</th>
<th>convergence acceleration for nonlinear optimization</th>
<th>convergence acceleration for nonlinear systems</th>
<th>convergence acceleration for nonlinear optimization</th>
</tr>
</thead>
<tbody>
<tr>
<td>P-CG</td>
<td>NCG, P?</td>
<td>NCG, P?</td>
<td>P-NCG</td>
</tr>
<tr>
<td>P-GMRES</td>
<td>?</td>
<td>P-NGMRES (Washio and Oosterlee, Anderson)</td>
<td>P-NGMRES for optimization</td>
</tr>
<tr>
<td>MG</td>
<td>?</td>
<td>FAS</td>
<td>adaptive AMG-FAS for optimization</td>
</tr>
</tbody>
</table>
(4) adaptive AMG-FAS for accelerating ALS

- with Killian Miller


\[
A = U \Sigma V^t = \sigma_1 u_1 v_1^T + \ldots + \sigma_n u_n v_n^T
\]

- SVD case: separate coarsening and interpolation for \(u\) and \(v\) variables

\[
A \in \mathbb{R}^{m \times n}, \quad A v = \sigma u, \quad u = P u_c, \quad P^t A Q v_c = \sigma P^t B P u_c,
\]
\[
A^t u = \sigma v, \quad v = Q v_c, \quad Q^t A^t P u_c = \sigma Q^t C Q v_c,
\]
adaptive AMG-FAS for accelerating ALS

\[ A = U \Sigma V^t = \sigma_1 u_1 v_1^T + \ldots + \sigma_n u_n v_n^T \]

- **SVD case:** separate coarsening and interpolation for \( u \) and \( v \) variables

  \[ A \in \mathbb{R}^{m \times n}, \quad A v = \sigma u, \quad u = P u_c, \quad P^t A Q v_c = \sigma P^t B P v_c, \]
  
  \[ A^t u = \sigma v, \quad v = Q v_c, \quad Q^t A^t P u_c = \sigma Q^t C Q v_c, \]

  (note: collective interpolation of \( u \) singular vectors by \( P, \ldots \))

- we don’t know the nature of the slow-to-converge components in advance: determine \( P, Q, R, \ldots \) adaptively using Bootstrap AMG in a ‘multiplicative’ setup phase (Brandt/Brannick/Kahl/Livshits and Kushnir/Galun/Brandt)

- use FAS in an ‘additive’ solve phase (Brandt/McCormick/Ruge and Borzi/Borzi)
canonical tensor decomposition: notation

- Decompose a tensor as a sum of $R$ rank-one terms
- For a third-order tensor:

$$\mathbf{X} \approx \sum_{r=1}^{R} \mathbf{a}_r \circ \mathbf{b}_r \circ \mathbf{c}_r = [\mathbf{A}, \mathbf{B}, \mathbf{C}]$$

Factor matrices:

$$\mathbf{A} = [\mathbf{a}_1, \ldots, \mathbf{a}_R], \quad \mathbf{B} = [\mathbf{b}_1, \ldots, \mathbf{b}_R], \quad \mathbf{C} = [\mathbf{c}_1, \ldots, \mathbf{c}_R]$$
some tensor machinery...

Mode-$n$ Matricization: $\mathcal{Z} \rightarrow \mathbf{Z}_{(n)}$

- Transforms a tensor into a matrix by reordering tensor elements

Khatri-Rao product

- For $\mathbf{A} \in \mathbb{R}^{I \times J}$ and $\mathbf{B} \in \mathbb{R}^{K \times J}$

\[
\mathbf{A} \odot \mathbf{B} = [\mathbf{a}_1 \otimes \mathbf{b}_1 \mathbf{a}_2 \otimes \mathbf{b}_2 \cdots \mathbf{a}_J \otimes \mathbf{b}_J] \in \mathbb{R}^{IK \times J}
\]
(4a) multiplicative phase: coarse equations

\[
\text{minimize } \quad f(A^{(1)}, \ldots, A^{(N)}) := \frac{1}{2} \left\| \mathcal{Z} - \left[ A^{(1)}, \ldots, A^{(N)} \right] \right\|^2
\]

- Fine-level equations are the gradient equations of \( f \):

\[
\mathcal{Z}_{(n)} \left( A^{(N)} \odot \cdots \odot A^{(n+1)} \odot A^{(n-1)} \odot \cdots \odot A^{(1)} \right) = A^{(n)} \Gamma^{(n)}
\]

- Assume that \( \text{range}(P^{(n)}) \) approximately contains \( A^{(n)} \), i.e.,

\[
A^{(n)} \approx P^{(n)} A_c^{(n)} \quad \text{for } n = 1, \ldots, N
\]

- Coarse-level equations:

\[
\mathcal{Z}_c^{(n)} \left( A_c^{(N)} \odot \cdots \odot A_c^{(n+1)} \odot A_c^{(n-1)} \odot \cdots \odot A_c^{(1)} \right) = B^{(n)} A_c^{(n)} \Gamma_c^{(n)}
\]

\[
B^{(n)} = P^{(n)T} \quad \text{(SPD)}
\]

\[
\mathcal{Z}_c = \mathcal{Z} \times_1 P^{(1)T} \times_2 P^{(2)T} \cdots \times_N P^{(N)T}
\]
multiplicative phase: coarse equations

- Compute Cholesky factorization: \( B^{(n)} = L^{(n)}L^{(n)\text{T}} \)
- Let \( \hat{A}_c^{(n)} = L^{(n)\text{T}}A_c^{(n)} \) for \( n = 1, \ldots, N \)
- Transformed coarse-level equations:

\[
\hat{Z}_c^{(n)} \left( \hat{A}_c^{(N)} \odot \cdots \odot \hat{A}_c^{(n+1)} \odot \hat{A}_c^{(n-1)} \odot \cdots \odot \hat{A}_c^{(1)} \right) = \hat{A}_c^{(n)}\Gamma_c^{(n)}
\]

\[
\hat{Z}^c = \mathcal{Z}_c \times_1 \hat{P}_1^{(1)\text{T}} \times_2 \hat{P}_2^{(2)\text{T}} \cdots \times_N \hat{P}_N^{(N)\text{T}}
\]

\[
\hat{P}_n^{(n)} = P_n^{(n)}L^{(n)\text{T}}, \quad \hat{R}_n^{(n)} = \hat{P}_n^{(n)\text{T}}
\]

- Transformed coarse-level equations are gradient equations of

\[
f_c(\hat{A}_c^{(1)}, \ldots, \hat{A}_c^{(N)}) := \frac{1}{2} \left\| \hat{Z}_c^c - \left[ \hat{A}_c^{(1)}, \ldots, \hat{A}_c^{(N)} \right] \right\|^2
\]
multiplicative phase: bootstrap V-cycle

- Build $\hat{P}^{(1)}, \ldots, \hat{P}^{(N)}$
- Build $\hat{Z}^c$

\[
A^{(n)} = \hat{P}^{(n)} \hat{A}_c^{(n)}, \quad n = 1, \ldots, N
\]
multiplicative phase: geometric coarsening

- Partition $\Omega_n = \{1, \ldots, I_n\}$ into coarse points $\mathcal{C}_n$ and fine points $\mathcal{F}_n$
- Standard geometric coarsening:

$$\mathcal{C}_n = \{\text{odd points in } \Omega_n\}, \quad \mathcal{F}_n = \{\text{even points in } \Omega_n\}$$

\[ p_{ij}^{(n)} = \begin{cases} w_{ij}^{(n)}, & i \in \mathcal{F}_n, \ j \in \{i/2, i/2 + 1\} \\ 1, & i \in \mathcal{C}_n, \ j = (i + 1)/2 \\ 0, & \text{otherwise,} \end{cases} \]

\[ P^{(n)} = \begin{bmatrix} 1 \\ \ast \\ \ast \\ 1 \\ \ast \\ \ast \\ 1 \end{bmatrix} \]
multiplicative phase: coarse equations

- Determine $\mathbf{P}^{(n)}$ via weighted least squares fitting of the test/boot factors, \textit{injected} to the coarse level
- Let $\mathbf{U} = \begin{bmatrix} \mathbf{A}_{t,1}^{(n)}, \ldots, \mathbf{A}_{t,n_t}^{(n)} | \mathbf{A}_{b}^{(n)} \end{bmatrix}$
- For each point $i \in \mathcal{F}_n$
  \[ \mu_k u_{ik} = \sum_{j \in \mathcal{C}_n^i} \mu_k w_{ij}^{(n)} (u_{k,c}^i)_j \]
  for $k = 1, \ldots, n_f = R(n_t + 1)$.

(One equation per test or boot factor)

- Choose $n_t$ so system is overdetermined
- Weights $\mu_k$ proportional to reciprocal of gradient norm
(4b) additive phase: FAS

- we apply the FAS to the set of nonlinear equations

\[ Z_{(n)} \left( A^{(N)} \odot \ldots \odot A^{(n+1)} \odot A^{(n-1)} \odot \ldots \odot A^{(1)} \right) = A^{(n)} \Gamma^{(n)} \]

using the coarse operators and interpolation matrices computed in the setup phase

- we apply block nonlinear Gauss-Seidel as the smoother on all levels
additive phase: FMG FAS
(4c) adaptive AMG numerical results

Finite Difference Laplacian Tensor

\((N = 4, s = 20, R = 6)\)

Black: ALS, Dashed: Multilevel, Gray: Multilevel+FMG

![Graphs showing gradient norm over iterations and time](image)
adaptive AMG numerical results

**IJK Tensor**

Third order tensor of size $s \times s \times s$ with elements

$$z_{ijk} = \left( i^2 + j^2 + k^2 \right)^{-1/2} \quad \text{for } i, j, k = 1, \ldots, s.$$  

$s = 100, R = 5$

Black: ALS, Gray: Multilevel+FMG

![Graphs showing gradient norm over iterations and time](Image)
### adaptive AMG numerical results

Third order tensor of size $s \times s \times s$ with elements

$$z_{i.i.k} = (i^2 + i^2 + k^2)^{-1/2} \quad \text{for} \quad i, i, k = 1, \ldots, s.$$ 

<table>
<thead>
<tr>
<th>problem parameters</th>
<th>ALS</th>
<th>Multilevel + FMG</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>it</td>
<td>time</td>
</tr>
<tr>
<td>$s = 50, R = 2$</td>
<td>161</td>
<td>0.7</td>
</tr>
<tr>
<td>$s = 50, R = 3$</td>
<td>2435</td>
<td>11.8</td>
</tr>
<tr>
<td>$s = 50, R = 4$</td>
<td>4838</td>
<td>26.1</td>
</tr>
<tr>
<td>$s = 50, R = 5$</td>
<td>301</td>
<td>30.0</td>
</tr>
<tr>
<td>$s = 100, R = 2$</td>
<td>253</td>
<td>10.7</td>
</tr>
<tr>
<td>$s = 100, R = 3$</td>
<td>1695</td>
<td>80.2</td>
</tr>
<tr>
<td>$s = 100, R = 4$</td>
<td>3836</td>
<td>202.2</td>
</tr>
<tr>
<td>$s = 100, R = 5$</td>
<td>7854</td>
<td>455.2</td>
</tr>
<tr>
<td>$s = 200, R = 2$</td>
<td>274</td>
<td>90.3</td>
</tr>
<tr>
<td>$s = 200, R = 3$</td>
<td>1830</td>
<td>682.3</td>
</tr>
<tr>
<td>$s = 200, R = 4$</td>
<td>2998</td>
<td>1249.5</td>
</tr>
<tr>
<td>$s = 200, R = 5$</td>
<td>5686</td>
<td>2611.4</td>
</tr>
</tbody>
</table>
adaptive AMG numerical results

**Fig. 6.6.** Random data problem. Convergence plot for test 5 in Table 6.5 ($s = 100, R = 5$). The solid black line is ALS and the dashed line is the multilevel method without FMG.
<table>
<thead>
<tr>
<th>convergence acceleration for linear systems</th>
<th>convergence acceleration for nonlinear optimization</th>
<th>convergence acceleration for nonlinear systems</th>
<th>convergence acceleration for nonlinear optimization</th>
</tr>
</thead>
<tbody>
<tr>
<td>P-CG</td>
<td>NCG, P?</td>
<td>NCG, P?</td>
<td>P-NCG</td>
</tr>
<tr>
<td>P-GMRES</td>
<td>?</td>
<td>P-NGMRES (Washio and Oosterlee, Anderson)</td>
<td>P-NGMRES for optimization</td>
</tr>
<tr>
<td>MG</td>
<td>?</td>
<td>FAS</td>
<td>adaptive AMG-FAS for optimization</td>
</tr>
</tbody>
</table>


thank you
step II: N-GMRES acceleration: \( \nabla f(A_R) = g(A_R) = 0 \)

- windowed implementation
- several possibilities to solve the small least-squares problems:
  - normal equations (with reuse of scalar products; cost \( 2nw \)) (Washio and Oosterlee, 1997)
  - QR with factor updating (Walker and Ni, 2011)
  - SVD and rank-revealing QR (Fang and Saad, 2009)

\[
\hat{\mathbf{u}}_{i+1} = \tilde{\mathbf{u}}_{i+1} + \sum_{j=0}^{i} \alpha_j (\tilde{\mathbf{u}}_{i+1} - \mathbf{u}_j)
\]

find coefficients \((\alpha_0, \ldots, \alpha_i)\) that minimize

\[
\|g(\tilde{\mathbf{u}}_{i+1}) + \sum_{j=0}^{i} \alpha_j (g(\tilde{\mathbf{u}}_{i+1}) - g(\mathbf{u}_j))\|_2.
\]