Tensor Computations and Applications in Data Mining

Lars Eldén

Department of Mathematics
Linköping University, Sweden
Joint work with Berkant Savas

SIAM AM July 2008
Murray & Rice, Differential geometry and statistics, 1993:

\[
\xi(x)^{i_1 \ldots i_r}_{j_1 \ldots j_s} = \xi(\theta)^{k_1 \ldots k_r}_{l_1 \ldots l_s} \frac{\partial x^{i_1}}{\partial \theta^{k_1}} \ldots \frac{\partial x^{i_r}}{\partial \theta^{k_r}} \frac{\partial \theta^{l_1}}{\partial x^{k_1}} \ldots \frac{\partial \theta^{l_s}}{\partial x^{k_s}}
\]  

(8.7.1)

Classically it would have been said that the tensor transforms by this rule. It is horrible formulae like this that have given tensor analysis a bad name.

“... the manipulation of matrices is a hundred times better supported in our brains and in our software tools than that of tensors.”

(N. Trefethen, Maxims about numerical mathematics, science, computers, and life on earth)
We need a notational and conceptual framework that

- exhibits the structure of the problems
- is independent of the order of the tensor, or easily generalizable
- allows the formulation and implementation of algorithms

Q: Can we find such a framework in math books on tensor calculus?
We need a notational and conceptual framework that

- exhibits the structure of the problems
- is independent of the order of the tensor, or easily generalizable
- allows the formulation and implementation of algorithms

Q: Can we find such a framework in math books on tensor calculus?
A: NO! (in general), because we are asking different questions now. Many fundamental mathematical problems are open!
We need a notational and conceptual framework that
- exhibits the structure of the problems
- is independent of the order of the tensor, or easily generalizable
- allows the formulation and implementation of algorithms

Q: Can we find such a framework in math books on tensor calculus?
A: NO! (in general), because we are asking different questions now. Many fundamental mathematical problems are open!

Tensor methods have been used since the 1960’s in psychometrics and chemometrics! Only recently in numerical community. Applications in signal processing and various areas of data mining.

Recent survey:
Example: Classification of hand-written digits

3-tensor $\mathcal{D}$ with
- pixel mode, 400 pixels
- digit mode, $\sim$ 1000 digits per class
- class mode, 10 classes

All digits of one class represented by a slice
1. Expansion in terms of rank-1 matrices:

\[ X = \sum_{i=1}^{n} \sigma_i u_i v_i^T = \sigma_1 u_1 v_1^T + \sigma_2 u_2 v_2^T + \cdots \]

2. Matrix decomposition: \( \mathbb{R}^{m \times n} \ni X = U \Sigma V^T \)
Tensor Expansion in Rank-1 Terms

$A = \sum \sum \sum + \cdots$

- **Parafac/Candecomp/Kruskal**: Harshman, Caroll, Chang 1970
- Numerous papers in psychometrics and chemometrics
- From a mathematical point of view: difficult problem, sometimes ill-posed, see De Silva and Lim 2006.
- From the point of view of applications: very useful! (Rasmus Bro’s talk)
Tensor Decomposition: Tucker Model

\[ A = U^{(1)} S U^{(2)} \]

- Tucker 1964, numerous papers in psychometrics and chemometrics
- De Lathauwer, De Moor, Vandewalle, SIMAX 2000: notation, theory.
- The matrices \( U^{(i)} \) are usually orthogonal.

This talk: Tucker model for 3-tensors only!
Classification of Handwritten Digits

"Model problem" in pattern recognition
HOSVD for Data Reduction

pixel mode, 400 pixels
digit mode, $\sim 1000$ digits per class
class mode, 10 classes

Cf. low-rank approximation of matrix by SVD: $A \approx U_k \Sigma_k V_k^T$
Project all Digits to Low Dimension

Each column is a digit in low dimension

Slice $\mu$ of $\mathcal{F}$ is a basis for class $\mu$

Compute the SVD of each slice: $\mathcal{F}(::, ::, \mu) = U^\mu \Sigma^\mu (V^\mu)^T$ and use $k$ columns, $U^\mu_k$, as basis vectors.
Classification with HOSVD Compression

- **Training phase:**
  1. Collect the training digits into a tensor $\mathcal{D}$.
  2. Compute the HOSVD of $\mathcal{D}$.
  3. Compute the low rank “basis” tensor $\mathcal{F} = (P^T)_1 \cdot \mathcal{D}$.
  4. Compute and store the basis matrices $B^\mu = U^\mu_k$ for each class.

- **Test phase:** For each test digit $d$
  1. Project $d = P^T d$.
  2. Compute the residuals $R(\mu) = \| (I - B^\mu (B^\mu)^T) d \|$, $\mu = 1, \ldots, 10$.
  3. Determine $\mu_{\text{min}} = \text{argmin}_\mu R(\mu)$ and classify $d$ as $\mu_{\text{min}}$. 
Figure: Error rates for different compressions (> 97.8%), and basis dimension.
Mode-1 Multiplication of a Tensor by a Matrix

Assume that dimensions are such that all operations are well-defined.
Mostly 3-tensors. Lim’s notation. (No standard notation yet)

\[ \mathcal{B} = (X) \cdot A, \quad \mathcal{B}(i, j, k) = \sum_{\nu=1}^{n} x_{i\nu} a_{\nu jk}. \]

All column vectors are multiplied by the matrix \(X\).
Multiplication in all modes at the same time:

\[ \mathcal{B} = (X, Y, Z) \cdot A, \quad \mathcal{B}(i, j, k) = \sum_{\nu, \mu, \lambda} x_{i\nu} y_{j\mu} z_{k\lambda} a_{\nu\mu\lambda}. \]

For convenience we write

\[ \mathcal{B} = (X^T, Y^T, Z^T) \cdot A = A \cdot (X, Y, Z) \]
Inner Product and Norm

Inner product (contraction: $\mathbb{R}^{n \times n \times n} \to \mathbb{R}$)

$$\langle A, B \rangle = \sum_{i,j,k} a_{ijk} b_{ijk}$$

The Frobenius norm:

$$\|A\| = \langle A, A \rangle^{1/2}$$

---

Matrix case

$$\langle A, B \rangle = \text{tr}(A^T B)$$
Partial Contractions

\[ C = \langle A, B \rangle_{1} , \quad D = \langle A, B \rangle_{1:2} , \quad e = \langle A, B \rangle = \langle A, B \rangle_{1:3} , \]

\[ c_{jklm} = \sum_{\lambda} a_{\lambda jk} b_{\lambda lm} , \quad (4\text{-tensor}), \]

\[ d_{jk} = \sum_{\lambda, \mu} a_{\lambda j\mu} b_{\lambda \mu k} , \quad (2\text{-tensor}), \]

\[ e = \sum_{\lambda, \mu, \nu} a_{\lambda \mu \nu} b_{\lambda \mu \nu} , \quad (\text{scalar}). \]

Notation (3-tensor):

\[ \langle A, B \rangle_{1:2} = \langle A, B \rangle_{-3} \]
Tensor SVD (HOSVD): \( \mathbf{A} = (\mathbf{U}^{(1)}, \mathbf{U}^{(2)}, \mathbf{U}^{(3)}) \cdot \mathbf{S} \)

1. Compute the SVD of all mode\(-i\) vectors
2. \( \mathbf{U}^{(i)} \) is left singular matrix of mode \( i \)
3. \( \mathbf{S} := \mathbf{A} \cdot (\mathbf{U}^{(1)}, \mathbf{U}^{(2)}, \mathbf{U}^{(3)}) \)

The “mass” of \( \mathbf{S} \) is concentrated around the \((1, 1, 1)\) corner.

Not optimal: does not give the solution of \( \min_{\text{rank}(\mathbf{B})=(r_1,r_2,r_3)} \| \mathbf{A} - \mathbf{B} \| \)

Best rank—\((r_1, r_2, r_3)\) approximation:

\[
\min_{X,Y,Z,S} \|A - (X, Y, Z) \cdot S\|,
\]

\[
X^T X = I, \quad Y^T Y = I, \quad Z^T Z = I
\]

The problem is over-parameterized!
Best Approximation

\[
\begin{align*}
\min_{\text{rank}(\mathcal{B})=(r_1,r_2,r_3)} & \quad \|A - \mathcal{B}\| \\
\end{align*}
\]

is equivalent to

\[
\begin{align*}
\max_{X,Y,Z} & \quad \Phi(X, Y, Z) = \frac{1}{2} \|A \cdot (X, Y, Z)\|^2 \\
& = \frac{1}{2} \sum_{j,k,l} \left( \sum_{\lambda,\mu,\nu} a_{\lambda\mu\nu} x_{\lambda j} y_{\mu k} z_{\nu l} \right)^2,
\end{align*}
\]

subject to

\[
\begin{align*}
X^T X &= I_{r_1}, & Y^T Y &= I_{r_2}, & Z^T Z &= I_{r_3}
\end{align*}
\]
The Frobenius norm is invariant under orthogonal transformations:

$$\Phi(X, Y, Z) = \Phi(XU, YV, ZW) = \frac{1}{2} \| A \cdot (XU, YV, ZW) \|^2$$

for orthogonal $U \in \mathbb{R}^{r_1 \times r_1}$, $V \in \mathbb{R}^{r_2 \times r_2}$, and $W \in \mathbb{R}^{r_3 \times r_3}$. Maximize $\Phi$ over equivalence classes

$$[X] = \{ XU \mid U \text{ orthogonal} \}.$$ 

Product of manifolds: $\text{Gr}^3 = \text{Gr}(J, r_1) \times \text{Gr}(K, r_2) \times \text{Gr}(L, r_3)$

$$\max_{(X,Y,Z) \in \text{Gr}^3} \Phi(X, Y, Z) = \max_{(X,Y,Z) \in \text{Gr}^3} \frac{1}{2} \langle A \cdot (X, Y, Z), A \cdot (X, Y, Z) \rangle$$
Newton’s Method on one Grassmann Manifold

Taylor expansion + linear algebra on tangent space\(^1\) at \(X\)

\[ G(X(t)) \approx G(X(0)) + \langle \Delta, \nabla G \rangle + \frac{1}{2} \langle \Delta, H(\Delta) \rangle, \]

Grassmann gradient:

\[ \nabla G = \Pi_X G_x, \quad (G_x)_{jk} = \frac{\partial G}{\partial x_{jk}}, \quad \Pi_X = I - XX^T \]

The Newton equation for determining \(\Delta\):

\[ \Pi_X \langle G_{xx}, \Delta \rangle_{1:2} - \Delta \langle X, G_x \rangle_1 = -\nabla G, \quad (G_{xx})_{jklm} = \frac{\partial^2 G}{\partial x_{jk} \partial x_{lm}}. \]

\(^1\)Tangent space at \(X\): all matrices \(Z\) satisfying \(Z^T X = 0\).
Newton-Grassmann Algorithm on $\text{Gr}^3$

Here: local coordinates

Given tensor $\mathcal{A}$ and starting points $(X_0, Y_0, Z_0) \in \text{Gr}^3$

repeat

1. compute the Grassmann gradient $\nabla \hat{\Phi}$
2. compute the Grassmann Hessian $\hat{H}$
3. matricize $\hat{H}$ and vectorize $\nabla \hat{\Phi}$
4. solve $D = (D_x, D_y, D_z)$ from the Newton equation
5. take a geodesic step along the direction $D$, giving new iterates $(X,Y,Z)$

until $\|\nabla \hat{\Phi}\|/\Phi < \text{TOL}$

Implementation using TensorToolbox (Bader/Kolda) and home-made object-oriented Grassmann classes in Matlab
Newton’s method on $\text{Gr}^3$

Differentiate $\Phi(X, Y, Z)$ along a geodesic curve $(X(t), Y(t), Z(t))$ in the direction $(\Delta_x, \Delta_y, \Delta_z)$:

$$\frac{\partial x_{st}}{\partial t} = (\Delta_x)_{st},$$

and

$$\left(\frac{dX(t)}{dt}, \frac{dY(t)}{dt}, \frac{dZ(t)}{dt}\right) = (\Delta_x, \Delta_y, \Delta_z),$$

Since $\mathcal{A} \cdot (X, Y, Z)$ is linear in $X, Y, Z$ separately:

$$\frac{d(\mathcal{A} \cdot (X, Y, Z))}{dt} = \mathcal{A} \cdot (\Delta_x, Y, Z) + \mathcal{A} \cdot (X, \Delta_y, Z) + \mathcal{A} \cdot (X, Y, \Delta_z).$$
First Derivative

\[
\frac{d\Phi}{dt} = \frac{1}{2} \frac{d}{dt} \langle \mathcal{A} \cdot (X, Y, Z), \mathcal{A} \cdot (X, Y, Z) \rangle = \langle \mathcal{A} \cdot (\Delta_x, Y, Z), \mathcal{A} \cdot (X, Y, Z) \rangle \\
+ \langle \mathcal{A} \cdot (X, \Delta_y, Z), \mathcal{A} \cdot (X, Y, Z) \rangle + \langle \mathcal{A} \cdot (X, Y, \Delta_z), \mathcal{A} \cdot (X, Y, Z) \rangle.
\]

We want to write \( \langle \mathcal{A} \cdot (\Delta_x, Y, Z), \mathcal{A} \cdot (X, Y, Z) \rangle \) in the form \( \langle \Delta_x, \Phi_x \rangle \)

Define the tensor \( \mathcal{F} = \mathcal{A} \cdot (X, Y, Z) \) and write

\[
\langle \mathcal{A} \cdot (\Delta_x, Y, Z), \mathcal{F} \rangle =: \langle \mathcal{K}_x(\Delta_x), \mathcal{F} \rangle = \langle \Delta_x, \mathcal{K}_x^{*}\mathcal{F} \rangle,
\]

For fixed \( Y \) and \( Z \) we have a linear operator:

\[
\Delta_x \mapsto \mathcal{K}_x(\Delta_x) = \mathcal{A} \cdot (\Delta_x, Y, Z)
\]
Adjoint Operator

Linear operator:

\[ \Delta_x \mapsto \mathcal{K}_x(\Delta_x) = A \cdot (\Delta_x, Y, Z) \]

with adjoint

\[ \langle \mathcal{K}_x(\Delta_x), \mathcal{F} \rangle = \langle \Delta_x, \mathcal{K}^*_x \mathcal{F} \rangle = \langle \Delta_x, \langle A \cdot (I, Y, Z), \mathcal{F} \rangle \rangle \]

where the partial contraction is defined

\[ \langle B, C \rangle_{-1}(i_1, i_2) = \sum_{\mu, \nu} b_{i_1 \mu \nu} c_{i_2 \mu \nu} \]
Grassmann Gradient

\( X \)-part: multiply by \( \Pi_x = I - XX^T \)

\[
\Pi_x \Phi_x = \Pi_x \langle A \cdot (I, Y, Z), F \rangle_{-1} \\
= \langle A \cdot (I, Y, Z), A \cdot (X, Y, Z) \rangle_{-1} - XX^T \langle A \cdot (I, Y, Z), F \rangle_{-1} \\
= \langle A \cdot (I, Y, Z), A \cdot (I, Y, Z) \rangle_{-1} X - X \langle F, F \rangle_{-1},
\]

Complete gradient (recall \( F = A \cdot (X, Y, Z) \)):

\[
\nabla \Phi = (\Pi_x \Phi_x, \Pi_y \Phi_y, \Pi_z \Phi_z),
\]

where

\[
\Pi_x \Phi_x = \langle A \cdot (I, Y, Z), A \cdot (I, Y, Z) \rangle_{-1} X - X \langle F, F \rangle_{-1} \\
\Pi_y \Phi_y = \langle A \cdot (X, I, Z), A \cdot (X, I, Z) \rangle_{-2} Y - Y \langle F, F \rangle_{-2} \\
\Pi_z \Phi_z = \langle A \cdot (X, Y, I), A \cdot (X, Y, I) \rangle_{-3} Z - Z \langle F, F \rangle_{-3}
\]
\[
\frac{d^2 \Phi}{dt^2} = \\
= \langle A \cdot (\Delta x, Y, Z), A \cdot (\Delta x, Y, Z) \rangle + \langle A \cdot (\Delta x, \Delta y, Z), A \cdot (X, Y, Z) \rangle \\
+ \langle A \cdot (\Delta x, Y, Z), A \cdot (X, \Delta y, Z) \rangle + \langle A \cdot (\Delta x, Y, \Delta z), A \cdot (X, Y, Z) \rangle \\
+ \langle A \cdot (\Delta x, Y, Z), A \cdot (X, Y, \Delta z) \rangle + \cdots ,
\]

plus 10 analogous terms.
Grassmann Hessian

\[ \mathcal{H}(\Delta) = (\Phi_{x^*}(\Delta), \Phi_{y^*}(\Delta), \Phi_{z^*}(\Delta)) : T^3 \mapsto T^3, \]

where

\[ \Phi_{x^*}(\Delta) = \mathcal{H}_{xx}(\Delta_x) + \mathcal{H}_{xy}(\Delta_y) + \mathcal{H}_{xz}(\Delta_z), \quad \Phi_{x^*}(\cdot) : T^3 \rightarrow T_X, \]
\[ \Phi_{y^*}(\Delta) = \mathcal{H}_{yx}(\Delta_x) + \mathcal{H}_{yy}(\Delta_y) + \mathcal{H}_{yz}(\Delta_z), \quad \Phi_{y^*}(\cdot) : T^3 \rightarrow T_Y, \]
\[ \Phi_{z^*}(\Delta) = \mathcal{H}_{zx}(\Delta_x) + \mathcal{H}_{zy}(\Delta_y) + \mathcal{H}_{zz}(\Delta_z), \quad \Phi_{z^*}(\cdot) : T^3 \rightarrow T_Z, \]
Grassmann Hessian, “Diagonal Part”

\[ \mathcal{H}_{xx}(\Delta x) = \prod_X \langle B_x, B_x \rangle_{-1} \Delta x - \Delta x \langle F, F \rangle_{-1}, \quad B_x = A \cdot (I, Y, Z), \]
\[ \mathcal{H}_{yy}(\Delta y) = \prod_Y \langle B_y, B_y \rangle_{-2} \Delta y - \Delta y \langle F, F \rangle_{-2}, \quad B_y = A \cdot (X, I, Z), \]
\[ \mathcal{H}_{zz}(\Delta z) = \prod_Z \langle B_z, B_z \rangle_{-3} \Delta z - \Delta z \langle F, F \rangle_{-3}, \quad B_z = A \cdot (X, Y, I). \]
Grassmann Hessian, “Upper Triangular Part”,

\[ \mathcal{H}_{xy}(\Delta y) = \prod_X \left( \left\langle \langle C_{xy}, \mathcal{F} \rangle_{-\{1,2\}}, \Delta y \right\rangle_{2,4;1,2} + \left\langle \langle B_x, B_y \rangle_{-\{1,2\}}, \Delta y \right\rangle_{4,2;1,2} \right), \]

where \( C_{xy} = A \cdot (I, I, Z) \), etc.

4-tensor contracted with a matrix giving a matrix:

\[ \left\langle \langle C_{xy}, \mathcal{F} \rangle_{-\{1,2\}}, \Delta y \right\rangle_{2,4;1,2} \]
Illustration of Hessian Computation

Local coordinates.
Methods for Best Approximation

- **Grassmann-based**
  1. Newton (LE, B. Savas)
  2. Trust region/Newton (Ishteva, De Lathauwer et al.)
  3. BFGS quasi-Newton (Savas, Lim)
  4. Limited memory BFGS (Savas, Lim)

- **Alternating**
  1. HOOI (Kroonenberg, De Lathauwer)
A random tensor $\mathcal{A} \in \mathbb{R}^{20 \times 20 \times 20}$ with random entries $N(0, 1)$ approximated with a rank $-(5, 5, 5)$ tensor.
A random tensor $\mathcal{A} \in \mathbb{R}^{100 \times 100 \times 100}$ with random entries $N(0, 1)$ approximated with a rank $-(5, 10, 20)$ tensor.
In information sciences the tensors are often sparse:

- Term-document-author (Dunlavy et al)
- Graphs, web link analysis (Kolda et al)

For **sparse matrices**: Krylov methods give low rank approximations:

\[ AV_k = U_k H_k \]

The matrix is only used as operator: \( u = Av \)
Can we generalize Krylov methods to tensors and obtain low rank approximations?
\( \beta_1 u_1 = b, \ v_0 = 0 \)

for \( i = 1 : k \)

\( \alpha_i v_i = A^T u_i - \beta_i v_{i-1}, \)

\( \beta_{i+1} u_{i+1} = Av_i - \alpha_i u_i \)

end

The coefficients \( \alpha_i \) and \( \beta_i \) are chosen to normalize the vectors.
\[ \beta_1 u_1 = b, \ v_0 = 0 \]

\textbf{for} \ i = 1 : k \\
\quad \alpha_i v_i = A^T u_i - \beta_i v_{i-1}, \ [\alpha_i v_i = A \cdot (u_i)_1 - \beta_i v_{i-1},] \\
\quad \beta_{i+1} u_{i+1} = Av_i - \alpha_i u_i \ [\beta_{i+1} u_{i+1} = A \cdot (v_i)_2 - \alpha_i u_i] \\
\textbf{end}

The coefficients \( \alpha_i \) and \( \beta_i \) are chosen to normalize the vectors.
Arnoldi style (i.e., including Gram-Schmidt orthogonalization)

- Let $u_1$ and $v_1$ be given
- $h_{111} w_1 = A \cdot (u_1, v_1)_{1,2}$

**for** $\nu = 2 : m$

- $h_u = A \cdot (U_{\nu-1}, v_{\nu-1}, w_{\nu-1})$
- $h_{\nu,\nu-1,\nu-1} u_\nu = A \cdot (v_{\nu-1}, w_{\nu-1})_{2,3} - U_{\nu-1} h_u$
- $h_\nu = A \cdot (u_\nu, V_{\nu-1}, w_{\nu-1})$
- $h_{\nu,\nu,\nu-1} v_\nu = A \cdot (u_\nu, w_{\nu-1})_{1,3} - V_{\nu-1} h_\nu$
- $h_w = A \cdot (u_\nu, v_\nu, W_{\nu-1})$
- $h_{\nu\nu\nu} w_\nu = A \cdot (u_\nu, v_\nu)_{1,2} - W_{\nu-1} h_w$

**end**

Approximate

$$A \approx (U_m, V_m, W_m) \cdot H, \quad H = \left(U_m^T, V_m^T, W_m^T\right) \cdot A$$
Krylov Method for Tensor Approximation

Arnoldi style (i.e., including Gram-Schmidt orthogonalization)

- Let \( u_1 \) and \( v_1 \) be given
- \( h_{111} w_1 = A \cdot (u_1, v_1)_{1,2} \)

\[ \text{for } \nu = 2 : m \]
\[
\begin{align*}
    h_u &= A \cdot (U_{\nu-1}, V_{\nu-1}, W_{\nu-1}) \\
    h_{\nu,\nu-1,\nu-1} u_\nu &= A \cdot (v_{\nu-1}, w_{\nu-1})_{2,3} - U_{\nu-1} h_u \\
    h_v &= A \cdot (u_\nu, V_{\nu-1}, w_{\nu-1}) \\
    h_{\nu,\nu,\nu-1} v_\nu &= A \cdot (u_\nu, w_{\nu-1})_{1,3} - V_{\nu-1} h_v \\
    h_w &= A \cdot (u_\nu, v_\nu, W_{\nu-1}) \\
    h_{\nu\nu\nu} w_\nu &= A \cdot (u_\nu, v_\nu)_{1,2} - W_{\nu-1} h_w
\end{align*}
\]

- end

Approximate

\[ A \approx (U_m, V_m, W_m) \cdot H, \quad H = (U^T_m, V^T_m, W^T_m) \cdot A \]
Many variants are possible: see the talk by Berkant Savas in the session **MS117** Friday at 4.30

Suitable for

- sparse tensors
- tensors whose dimensions vary rapidly (new data)
Conclusions

- Tensor methods/algorithms without index-wrestling
  - Indices hidden using matrix-inspired notation and object-oriented software
  - Generalization to higher order tensors is straightforward
  - Partial contractions play the role of adjoints
Conclusions

- Tensor methods/algorithms without index-wrestling
  - Indices hidden using matrix-inspired notation and object-oriented software
  - Generalization to higher order tensors is straightforward
  - Partial contractions play the role of adjoints
- Grassmann optimization (for Tucker model)
  - Needed because tensors cannot be deflated like matrices
  - Unconstrained optimization
  - Newton: Quadratic convergence
Conclusions

- Tensor methods/algorithms without index-wrestling
  - Indices hidden using matrix-inspired notation and object-oriented software
  - Generalization to higher order tensors is straightforward
  - Partial contractions play the role of adjoints

- Grassmann optimization (for Tucker model)
  - Needed because tensors cannot be deflated like matrices
  - Unconstrained optimization
  - Newton: Quadratic convergence

- Sparse tensors: Krylov methods
Conclusions

- Tensor methods/algorithms without index-wrestling
  - Indices hidden using matrix-inspired notation and object-oriented software
  - Generalization to higher order tensors is straightforward
  - Partial contractions play the role of adjoints

- Grassmann optimization (for Tucker model)
  - Needed because tensors cannot be deflated like matrices
  - Unconstrained optimization
  - Newton: Quadratic convergence

- Sparse tensors: Krylov methods

- Many fundamental mathematical and algorithmic problems remain
Conclusions

- Tensor methods/algorithms without index-wrestling
  - Indices hidden using matrix-inspired notation and object-oriented software
  - Generalization to higher order tensors is straightforward
  - Partial contractions play the role of adjoints
- Grassmann optimization (for Tucker model)
  - Needed because tensors cannot be deflated like matrices
  - Unconstrained optimization
  - Newton: Quadratic convergence
- Sparse tensors: Krylov methods
- Many fundamental mathematical and algorithmic problems remain
- Numerous new applications in information sciences

Lars Eldén (Linköping Univ.)
Tensor Computations
SIAM AM July 2008
Conclusions

Tensor methods/algorithms without index-wrestling
- Indices hidden using matrix-inspired notation and object-oriented software
- Generalization to higher order tensors is straightforward
- Partial contractions play the role of adjoints

Grassmann optimization (for Tucker model)
- Needed because tensors cannot be deflated like matrices
- Unconstrained optimization
- Newton: Quadratic convergence

Sparse tensors: Krylov methods

Many fundamental mathematical and algorithmic problems remain

Numerous new applications in information sciences

Tensor algorithms and computations can be (easily) managed if we define the right abstractions!
J. D. Carroll and J. J. Chang.  
Analysis of individual differences in multidimensional scaling via an n-way generalization of Eckart-Young decomposition.  

L. De Lathauwer, B. De Moor, and J. Vandewalle.  
A multilinear singular value decomposition.  

L. De Lathauwer, B. De Moor, and J. Vandewalle.  
On the best rank-1 and rank-$(R_1, R_2, \ldots, R_N)$ approximation of higher-order tensor.  

V. de Silva and L.-H. Lim.  
Tensor rank and the ill-posedness of the best low-rank approximation problem.  

Daniel M. Dunlavy, Tamara G. Kolda, and W. Philip Kegelmeyer.  
Multilinear algebra for analyzing data with multiple linkages.  
L. Eldén and B. Savas.
A Newton–Grassmann method for computing the best multi-linear rank-$(r_1, r_2, r_3)$ approximation of a tensor.
Submitted to SIMAX.

R. Harshman.
Foundations of the PARAFAC procedure: Models and conditions for an explanatory multi-modal factor analysis.

T. Kolda, B. Bader, and J. Kenny.
Higher-order web link analysis using multilinear algebra.

T. G. Kolda and B. W. Bader.
Tensor decompositions and applications.

B. Savas and L. Eldén.
Handwritten digit classification using higher order singular value decomposition.
A. Smilde, R. Bro, and P. Geladi.  
*Multi-way Analysis: Applications in the Chemical Sciences.*  

L. R. Tucker.  
The extension of factor analysis to three-dimensional matrices.  

L. R. Tucker.  
Some mathematical notes on three-mode factor analysis.  