Tutorial on MATLAB for tensors and the Tucker decomposition

Tamara G. Kolda and
Brett W. Bader

Workshop on Tensor Decomposition and Applications
CIRM, Luminy, Marseille, France
August 29, 2005

Sandia is a multiprogram laboratory operated by Sandia Corporation, a Lockheed Martin Company,
for the United States Department of Energy's National Nuclear Security Administration
under contract DE-AC04-94AL85000.

Outline

- Part I
  - Basics of N-way arrays
    - Creating a tensor in MATLAB
    - Tensor multiplication
    - Matricizing a tensor
  - Tensor decompositions
    - What is the SVD of a tensor?
    - PARAFAC model
    - Tucker model
- Part II
  - Computing tensor decompositions
    - Rank-1 approximation
    - Tucker via HO-SVD
    - Tucker via ALS
  - The core array in the Tucker decomposition
  - Working with sparse tensors in MATLAB

MATLAB Tensor Classes

- Enables working with tensors in their native format
  - As tensors rather than as matrices or vectors
- Previewed at last year's workshop in Palo Alto
  - Lots of helpful suggestions from the participants
- Since submitted for publication
  - More helpful suggestions from the referees
- See also the N-way toolbox by Andersson and Bro (2000)

- We provide 4 MATLAB classes
  - **`tensor`**
  - **`tensor_as_matrix`**
  - **`cp_tensor`**
  - **`tucker_tensor`**
- Functions that are part of these classes will be highlighted in red

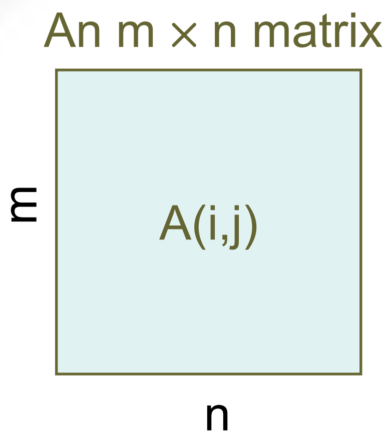
Download newest MATLAB tensor class and paper at:
http://csmr.ca.sandia.gov/~tgkolda/pubs/index.html#SAND2004-5187

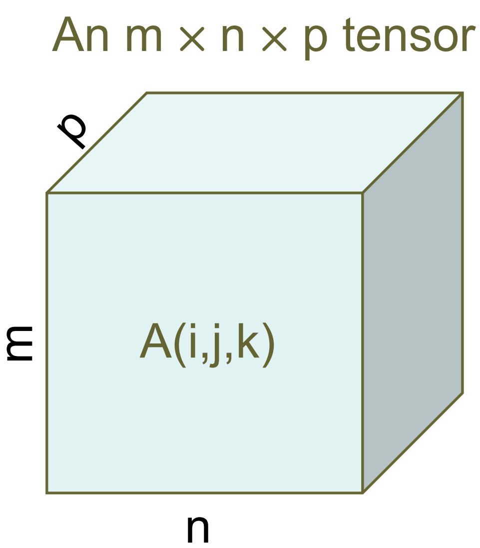
Unzip and add directory to MATLAB path via addpath command.

Sandia National Laboratories

The Basics

A tensor is a multidimensional array



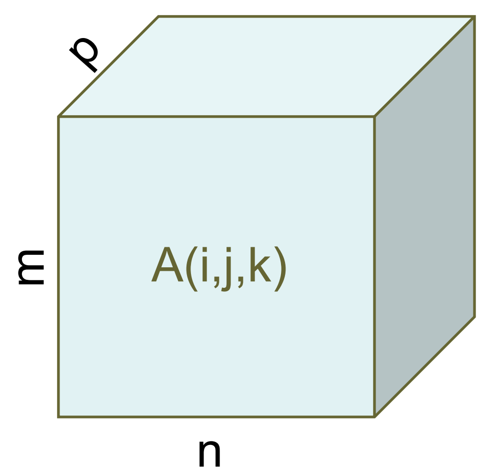


- Other names for tensors…
  - Multi-way array
  - N-way array
- Applications of SVD-like tensor decompositions
  - Image Compression
  - Data Mining
  - Psychometrics
  - Chemometrics
- The order of a tensor is the number of dimensions (a.k.a. ways or modes)
  - Scalar = tensor of order 0
  - Vector = tensor of order 1
  - Matrix = tensor of order 2

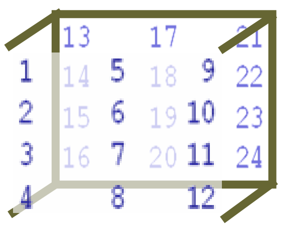
Sandia National Laboratories

Creating a tensor in MATLAB

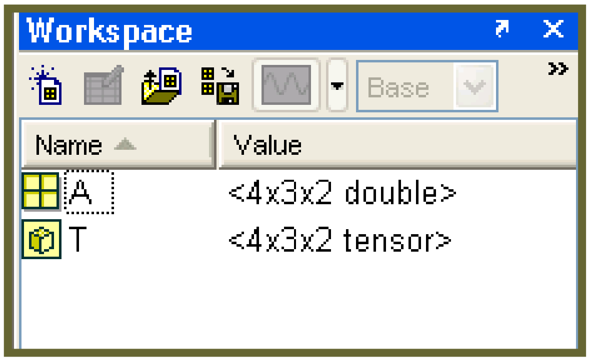
An m × n × p tensor



- MATLAB natively stores tensors as multidimensional arrays (MDAs)
  - `>> A = reshape(1:24, [4 3 2]);`
- The **tensor** class extends the MDA capabilities
  - `>> T = tensor(A) % convert`
  - ```
    T is a tensor of size 4 x 3 x 2
    T.data =
    (:,:,1) =
         1     5     9
         2     6    10
         3     7    11
         4     8    12
    (:,:,2) =
        13    17    21
        14    18    22
        15    19    23
        16    20    24
    ```
  - `>> T + T`
  - `>> T .* T`



Symbol is apropos!



Sandia National Laboratories
    ```

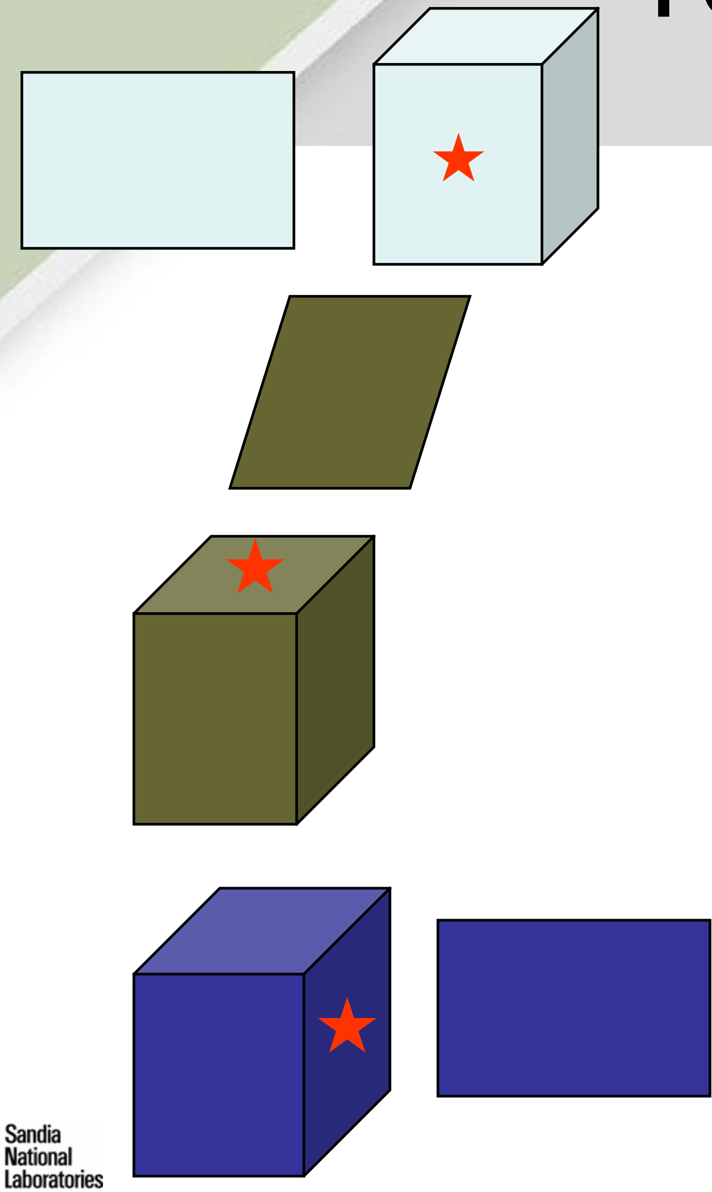
Aside: Trailing singleton dimensions

- An MDA does not explicitly track *trailing* singleton dimensions
  - Only non-singleton trailing dimensions are used in the calculation of **`ndims`**
  - Extra trailing singleton dimensions are added as needed, e.g., for **`size`**
- The **`tensor`** object, on the other hand, explicitly tracks its size, including trailing singleton dimensions
  - Constructor needs explicit size if there are trailing dimensions

```
>> A = rand(4,3,2,1);
>> ndims(A)
3
>> size(A,5)
1
>> T = tensor(A);
>> ndims(T)
3
T = tensor(A,[4,3,2,1]);
>> ndims(T)
4
>> size(T,5)
error
```

Sandia National Laboratories

Tensor Multiplication



- Two important special cases
  - Tensor-Matrix
  - Tensor-Vector
- General
  - Tensor-Tensor

From Kruskal (1977) for 3rd order tensors:

Now we introduce multiplication of an array by a matrix, whose product in general is an array. Since an array has three indices, it can be multiplied from three sides: we shall write

$$UX, \qquad \overset{V}{X}, \qquad XW$$

to indicate the three kinds of products: these indicate

$$\sum_i u_{i'i} x_{ijk}, \qquad \sum_j v_{j'j} x_{ijk}, \qquad \sum_k w_{k'k} x_{ijk}.$$

Sandia National Laboratories

Tensor-Matrix Multiplication

- Let A be an m x n x p tensor
- Let U be a q x m matrix
- Let V be a r x n matrix
- Mode-n multiplication means to multiply the matrix times the tensor in the n^th mode

$$B = A \times_1 U$$

$$(B)_{\ell jk} = \sum_{i=1}^{m} A_{ijk} U_{\ell i}$$

Alternate Notation: $B = A \bullet_1 U$

- `>> A = tensor(rand(4,3,2));`
- `>> U = rand(3,4);`

$$B = A \times_1 U$$

- `>> B = ttm(A,U,1);`
- `>> size(B)`
- `3 3 2` (size q x n x p)
- `>> V = rand(2,3);`

$$B = A \times_1 U \times_2 V$$

- `>> B = ttm(A,{U,V},[1,2]);`
- `>> size(B)`
- `3 2 2` (size q x r x p)

Sandia National Laboratories

Tensor-Vector Multiplication

- Let A be an m x n x p tensor
- Let u be an m-vector
- Let v be an n-vector

$$B = A \bar{\times}_1 u$$

$$(B)_{jk} = \sum_{i=1}^{m} A_{ijk} u_i$$

<u>Note</u>: The same symbol is often used for both tensor-matrix and tensor-vector.

- `>> A = tensor(rand(4,3,2));`
- `>> u = rand(4,1);`

$$B = A \bar{\times}_1 u$$

- `>> B = ttv(A,u,1);`
- `>> size(B)`
- `3 2` (size n x p)
- `>> v = rand(3,1);`

$$B = A \bar{\times}_1 u \bar{\times}_2 v$$

- `>> B = ttv(A,{u,v},[1,2]);`
- `>> size(B)`
- `2` (size p)

Sandia National Laboratories

Sequence Multiplication Notation

- We often want to compute a tensor A times a sequence of matrices (or vectors) in all modes but one
- E.g., Multiply a 4th order tensor in all modes but the 3rd
  - $A \times_1 U_1 \times_2 U_2 \times_4 U_4$
  - $A \times_{-3} \{U\}$
- We can do the same with vectors as well
  - $A \bar{\times}_1 u_1 \bar{\times}_2 u_2 \bar{\times}_4 u_4$
  - $A \bar{\times}_{-3} \{u\}$
- Notation is also useful in algorithm descriptions

- Create a tensor and some matrices
  - `>> A = tensor(rand(4,3,2,2));`
  - `>> for i = 1:4, U{i} = rand(2,size(A,i)); end;`
- The following produce equivalent results
  - `>> ttm(A, {U{1}, U{2}, U{4}},[1 2 4])`
  - `>> ttm(A,U,[1 2 4])`
  - `>> ttm(A, U, -3)`

Sandia National Laboratories

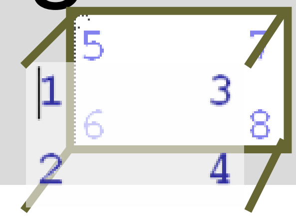
Tensor-Tensor Multiplication

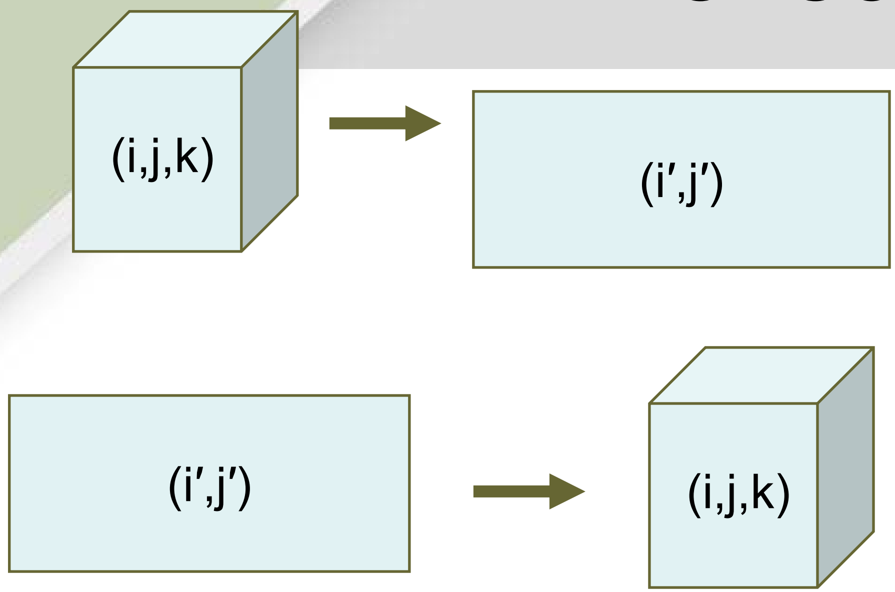
- Let A be an 4 x 2 x 3 tensor of all ones
  - `>> A = tensor(ones(4,2,3))`
- Let B be an 3 x 4 x 2 tensor
  - `>> B = tensor(rand(3,4,2))`
- Inner Product
  - `>> R = ttt(A,A,1:3)`
  - `24` (scalar)
- Outer product
  - `>> R = ttt(A,B);`
  - `>> size(R)`
  - `4 2 3 3 4 2`
- General multiplication along a subset of dimensions
  - `>> R = ttt(A,B,[1,3], [2,1]);`
  - `>> size(R)`
  - `2 2`

Aside: `ttt` is the kernel for `ttm` and `ttv`.

Sandia National Laboratories

Matricize: Converting a Tensor to a Matrix



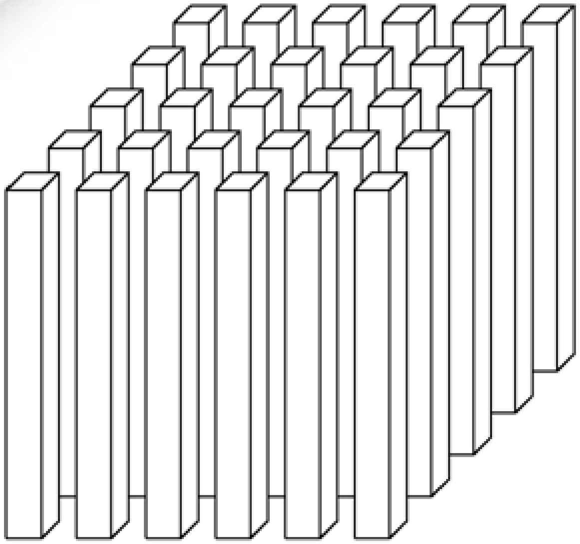


- Important Special Case
  - $A_{(n)}$ = matrix form of a tensor where the n^{th} dimension maps to the rows of the matrix
  - Also called “unfolding”
- Reverse matricize is also possible, provided that we know the mapping

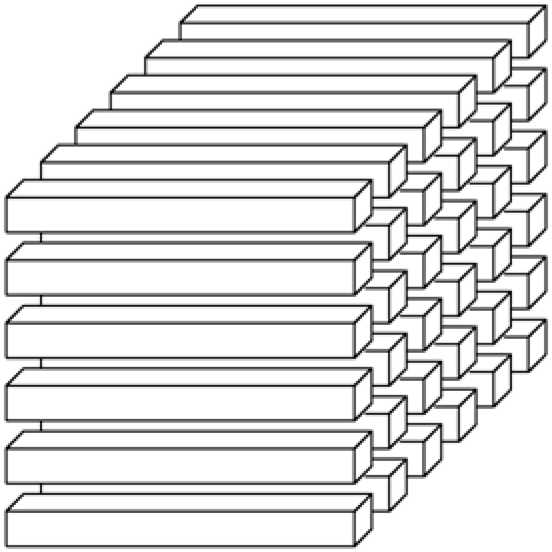
```
>> A=tensor(reshape(1:8,
   [2 2 2]));
>> tensor_as_matrix(A,1)
     1     3     5     7
     2     4     6     8
>> tensor_as_matrix(A,2)
     1     2     5     6
     3     4     7     8
>> tensor_as_matrix(A,3)
     1     2     3     4
     5     6     7     8
>> tensor_as_matrix(A,[1 2]);
   % Try for fun…
```

Tensor “Fibers”

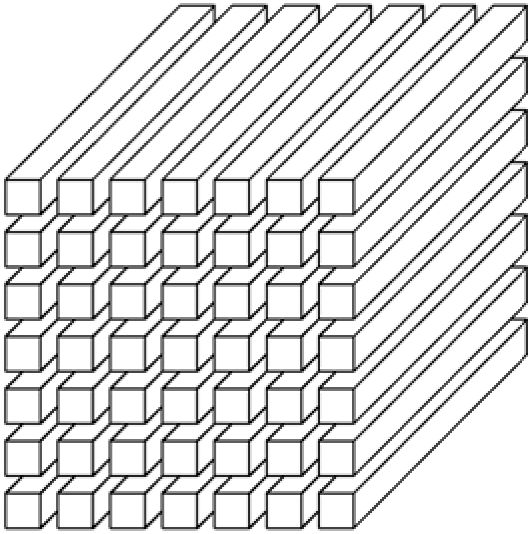
Another view of matricization is lining up the tensor fibers from a particular dimension.



Column fibers comprise columns of $A_{(1)}$



Row fibers comprise columns of $A_{(2)}$



Tube fibers comprise columns of $A_{(3)}$

Sandia National Laboratories

Why Matricize?

$$B = A \times_n U$$

$$B_{(n)} = U A_{(n)}$$

- Few software products support n-way arrays
- E.g., MATLAB added MDAs only recently
- Representation as a matrix may be the only option
  - In fact, our `ttt` operator converts the tensors to matrices to do the multiplication
- *However, rearranging the elements for different matricizations can be expensive and may be unnecessary*

Two Useful Facts for n-Mode Multiplication

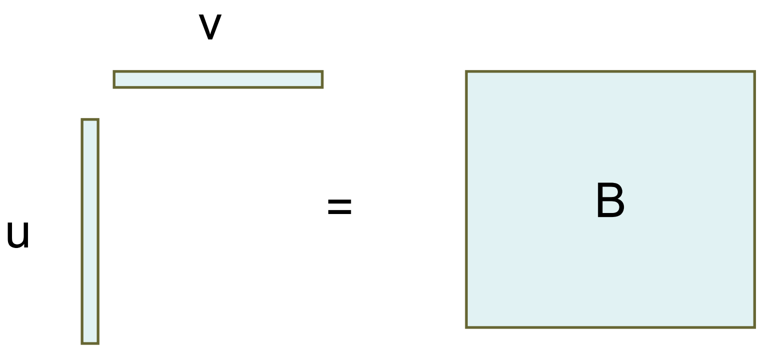
- Identity 1: $A \times_n (UV) = A \times_n V \times_n U$
- Identity 2: Let U be a p x q matrix with full column rank. If $B = A \times_n U$ then $A = B \times_n Z$ where Z is the q x p left inverse of U
- Example: If U has orthonormal columns, then $B = A \times_n U \Rightarrow A = B \times_n U^T$
- Proofs follow immediately from $B = A \times_n U \Leftrightarrow B_{(n)} = U A_{(n)}$

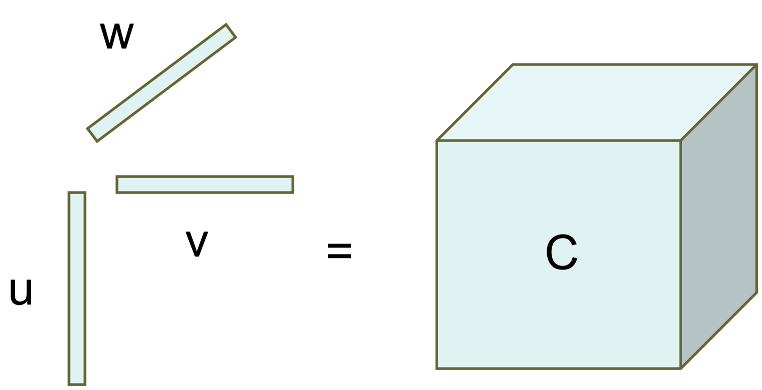
Sandia National Laboratories

Tensor Decompositions

Building Blocks: Rank-1 Tensors

- 2D
  - $B = u \circ v = u\, v^T$
  - $B(i,j) = u(i) \cdot v(j)$
- 3D (a.k.a. triad)
  - $C = u \circ v \circ w$
  - $C(i,j,k) = u(i) \cdot v(j) \cdot w(k)$
- 4D (a.k.a. 4-ad or N-ad)
  - $D = u \circ v \circ w \circ x$
  - $D(i,j,k,l) = u(i) \cdot v(j) \cdot w(k) \cdot x(l)$
- Alternate notation:
  - $u \otimes v$ or $u \Delta v$





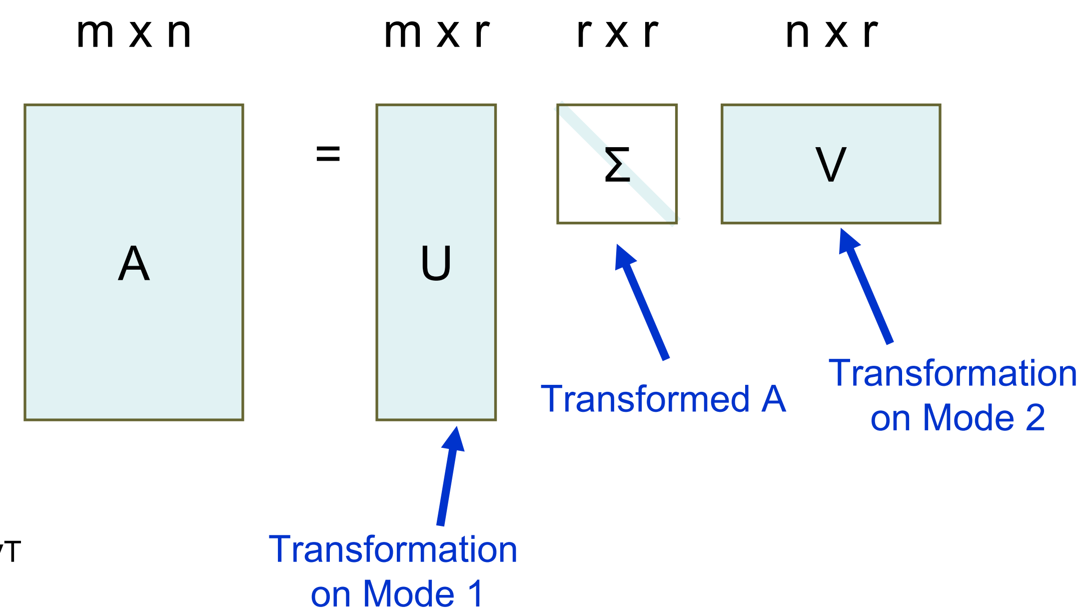
Sandia National Laboratories

Matrix SVD:
$A = \Sigma \times_1 U \times_2 V$

i^{th} column of U

$$A = \sum_{i=1}^{r} \sigma_{ii}\, u_i \circ v_i$$

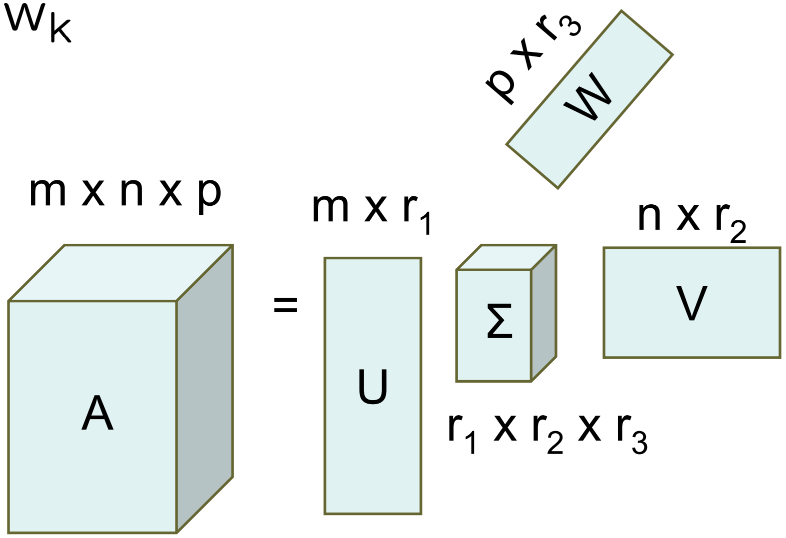
i^{th} column of V



- SVD or PCA: $A = U \Sigma V^T$
- Σ is a diagonal matrix
- Columns of U, V are orthonormal
- Two-way PCA: does not always require Σ to be diagonal (e.g., Henrion '94)
- ICA: U and V need not be be orthonormal

Tensor SVD: $A = \Sigma \times_1 U \times_2 V \times_3 W$

$$A = \sum_{i=1}^{r_1} \sum_{j=1}^{r_2} \sum_{k=1}^{r_3} \sigma_{ijk}\, u_i \circ v_j \circ w_k$$



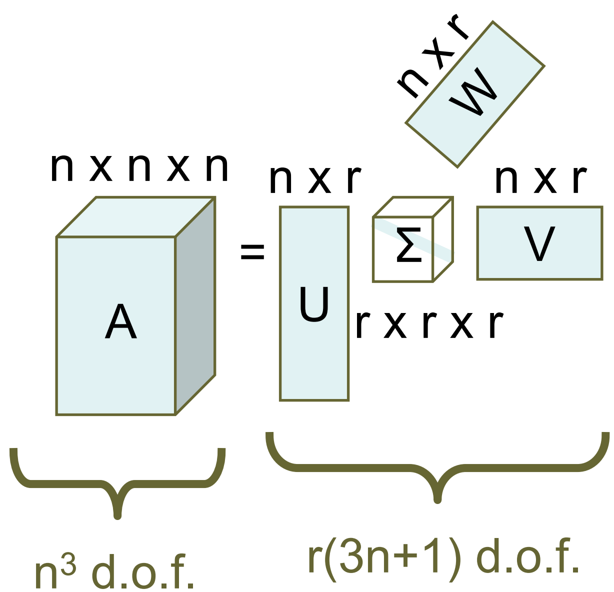
- Σ is the core tensor
- U, V, W are the components or factors
- In general, can have either
  - Diagonal core **or**
  - Orthonormal columns in components

Why can't we have both?

- Let A be a tensor of size n x n x n
- Suppose we do have both
  - diagonal Σ
  - orthogonal U,V,W
- Diagonal Σ yields

$$A = \sum_{i=1}^{r} \sigma_{iii}\, u_i \circ v_i \circ w_i$$

- Orthogonal U,V,W gives $r \leq n$
- $\Rightarrow$ Right-hand-side has only has $O(n^2)$ d.o.f.

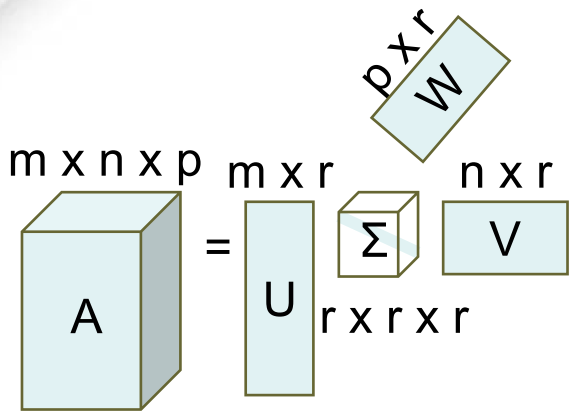


Two Choices

1. Diagonal Σ **or**
2. Orthogonal U,V,W

Diagonal Σ

CANDECOMP-PARAFAC (CP) Decomposition



$$A = \Sigma \times_1 U \times_2 V \times_3 W$$

$$A = \sum_{i=1}^{r} \sigma_{iii}\, u_i \circ v_i \circ w_i$$

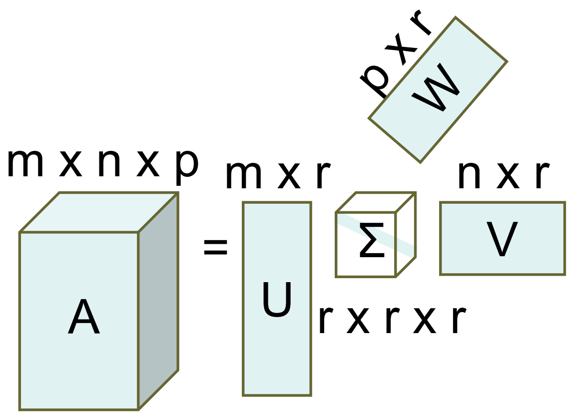
- CANDECOMP = Canonical Decomposition of Carroll and Chang (1970)
- PARAFAC = Parallel Factors of Harshman (1970)
- a.k.a. Tensor Canonical Decomposition
- Columns of U, V, and W are not orthogonal
- If r is *minimal*, then r is called the rank of the tensor (Kruskal 1977)
- Can have rank(A) > min{m,n,p}
- Often guaranteed to be a unique rank decomposition!
- Abbreviation CP follows Kiers (2000)

Sandia National Laboratories

CP Tensors in MATLAB

$$A = \Sigma \times_1 U \times_2 V \times_3 W$$

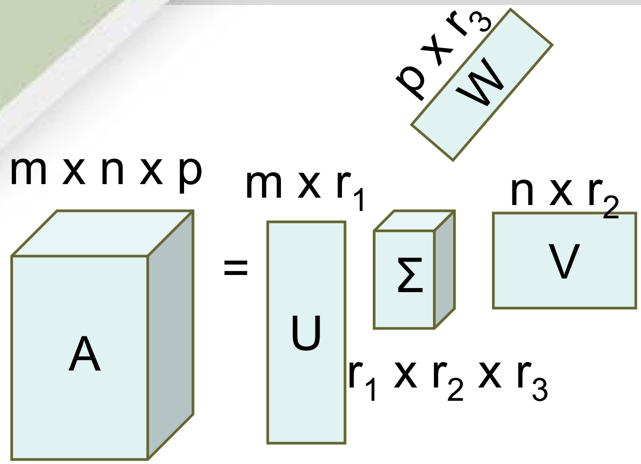
$$A = \sum_{i=1}^{r} \sigma_{iii}\, u_i \circ v_i \circ w_i$$




```
>> r = 2;
>> U = rand(2,r);
>> V = rand(3,r);
>> W = rand(2,r);
>> A = cp_tensor({U,V,W})
A is a CP tensor of size 2 x 3 x 2
A.lambda =
    1
    1
A.U{1} =
    0.5485    0.5973
    0.2618    0.0493
A.U{2} =
    0.5711    0.7400
    0.7009    0.4319
    0.9623    0.6343
A.U{3} =
    0.0839    0.9159
    0.9455    0.6020
```

U,V,W Orthogonornal

Tucker Decomposition



$$A = \Sigma \times_1 U \times_2 V \times_3 W$$

$$A = \sum_{i=1}^{r_1} \sum_{j=1}^{r_2} \sum_{k=1}^{r_3} \sigma_{ijk}\, u_i \circ v_j \circ w_k$$

- Proposed by Tucker (1966)
- a.k.a. three-mode factor analysis, three-mode PCA, orthogonal array decomposition
- Sum of all possible combination of vectors from U,V,W
- Σ is <u>not</u> diagonal
- Not unique

Σ is not free:

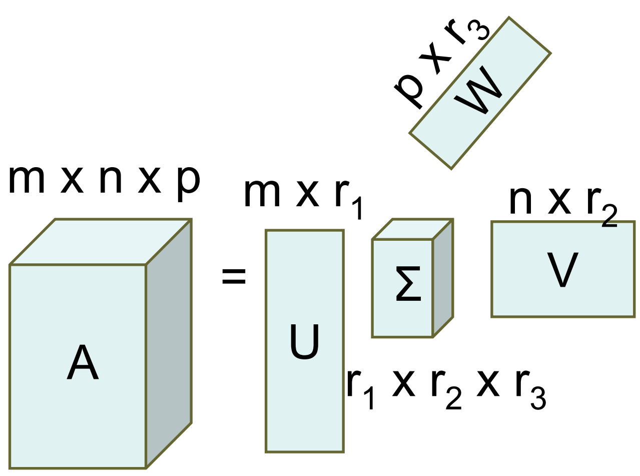
$$\Sigma = A \times_1 U^T \times_2 V^T \times_3 W^T$$

Sandia National Laboratories

Tucker Tensors in MATLAB

$$A = \Sigma \times_1 U \times_2 V \times_3 W$$

$$A = \sum_{i=1}^{r_1} \sum_{j=1}^{r_2} \sum_{k=1}^{r_3} \sigma_{ijk}\, u_i \circ v_j \circ w_k$$




```
>> siz = [2 3 2];
>> core = tensor(rand(2,2,1),[2 2 1]);
>> % Pick orthogonal matrices
>> for i = 1:3, [U{i},junk] =
    qr(rand(siz(i), size(core,i)),0);
    end
>> A = tucker_tensor(core,U)
A is a Tucker tensor of size 2 x 3 x 2
A.lambda is a tensor of size 2 x 2 x 1
A.lambda.data =
    0.2009    0.6262
    0.2731    0.5369
A.U{1} =
    0.0595    0.2713
    0.0890    0.4091
A.U{2} =
    0.4740    0.3290
    0.9090    0.4782
    0.5962    0.5972
A.U{3} =
    0.1614
    0.8295
```

Sandia National Laboratories

MATLAB Notes for Tucker and CP

- Access the core array (a vector for CP and a tensor for Tucker)
  - `>> A.lambda`
  - ```
    ans is a tensor of size 2 x 2 x 1
    ans.data =
        0.2009    0.6262
        0.2731    0.5369
    ```
- Access the n^th matrix
  - `>> A.U{2}`
  - ```
    ans =
        0.4740    0.3290
        0.9090    0.4782
        0.5962    0.5972
    ```
- Convert to a tensor object
  - `>> full(A)`
  - ```
    ans is a tensor of size 2 x 3 x 2
    ans.data =
    (:,:,1) =
        0.0163    0.0267    0.0259
        0.0245    0.0403    0.0390
    (:,:,2) =
        0.0837    0.1374   0.1332
        0.1261    0.2069    0.2005
    ```

Sandia National Laboratories
    ```

End of Part I

Questions?

Computing Tensor Decompositions

Focus is on Tucker – PARAFAC is left to Richard Harshman's tutorial later today.

Sandia National Laboratories

Rank-1 Approximation

- Goal: Find vectors u,v,w that minimize

$$\|A - \lambda \, \mathsf{u} \circ \mathsf{v} \circ \mathsf{w}\|_F^2 = \sum_{ijk} \left(\mathsf{A}_{ijk} - \lambda \, \mathsf{u}_i \mathsf{v}_j \mathsf{w}_k \right)^2$$

- WLOG, assume ‖ u ‖ = ‖ v ‖ = ‖ w ‖ = 1, and

$$\lambda = \mathsf{A} \,\bar{\times}_1\, \mathsf{u} \,\bar{\times}_2\, \mathsf{v} \,\bar{\times}_3\, \mathsf{w}$$

- Can repeatedly compute the best rank-1 approximation to the residual to get a “greedy PARAFAC” model

Sandia National Laboratories

Best Rank-1 Approximation

Optimal answer for fixed v & w

For k = 1, 2, . . .

$$\tilde{u}^{(k+1)} = A \bar{\times}_2 v^{(k)} \bar{\times}_3 w^{(k)}$$
$$\tilde{v}^{(k+1)} = A \bar{\times}_1 \tilde{u}^{(k+1)} \bar{\times}_3 w^{(k)}$$
$$\tilde{w}^{(k+1)} = A \bar{\times}_1 \tilde{u}^{(k+1)} \bar{\times}_2 \tilde{v}^{(k+1)}$$

$$u^{(k+1)} = \tilde{u}^{(k+1)} / \|\tilde{u}^{(k+1)}\|$$
$$v^{(k+1)} = \tilde{v}^{(k+1)} / \|\tilde{v}^{(k+1)}\|$$
$$w^{(k+1)} = \tilde{w}^{(k+1)} / \|\tilde{w}^{(k+1)}\|$$

$$\lambda^{(k+1)} = A \bar{\times}_1 u^{(k+1)} \bar{\times}_2 v^{(k+1)} \bar{\times}_3 w^{(k+1)}$$

end

Given Tensor A.
Find best rank-1 approximation:
$\lambda\, u \circ v \circ w$

Sandia National Laboratories

Higher Order Power Method: De Lathauwer, De Moor, Vandewalle (2000)

Power Method in MATLAB

```
% A = 3-way tensor
for i = 1:maxits
  u = ttv(A,{v,w},-1);
  v = ttv(A,{u,w},-2);
  w = ttv(A,{u,v},-3);
  u = u / norm(u);
  v = v / norm(v);
  w = w / norm(w);
  s = ttv(A,{u,v,w});
end
B = cp_tensor(s,{u,v,w})
C = tucker_tensor(tensor(s,[1 1 1]),{u,v,w});
```

See **examples/hopm.m** for full method.

Sandia National Laboratories

HO-SVD (Tucker1)

Given Tensor A.
Find a rank-(r_1,r_2,r_3) Tucker Decomposition

$\Sigma \times_1 U \times_2 V \times_3 W$

U has r_1 columns
V has r_2 columns
W has r_3 columns

$$
\begin{aligned}
U &= r_1 \text{ left singular vectors of } A_{(1)} \\
V &= r_2 \text{ left singular vectors of } A_{(2)} \\
W &= r_3 \text{ left singular vectors of } A_{(3)} \\
\Sigma &= A \times_1 U^T \times_2 V^T \times_3 W^T
\end{aligned}
$$

Sandia National Laboratories

Tucker (1966), De Lathauwer, De Moor, Vandewalle (2000)

MATLAB HO-SVD

```
% A = tensor
% r1,r2,r3 = desired ranks
[U,xx,zz] = svds(double(tensor_as_matrix(A,1)),r1)
[V,xx,zz] = svds(double(tensor_as_matrix(A,2)),r2)
[W,xx,zz] = svds(double(tensor_as_matrix(A,3)),r3)
S = ttm(A,{U',V',W'})
B = tucker_tensor(S,{U,V,W});
```

Best Rank-(r_1,r_2,r_3) Approximation

Given Tensor A.
Find best approximation:

$\Sigma \times_1 U \times_2 V \times_3 W$

U, V, W orthogonal
U has r_1 columns
V has r_2 columns
W has r_3 columns

Optimal answer for fixed V & W

For $k = 1, 2, \ldots$

$$X^{(k+1)} = A \times_2 V^{(k)^T} \times_3 W^{(k)^T}$$
$$U^{(k+1)} = \text{first } r_1 \text{ left singular vectors of } X_{(1)}^{(k+1)}$$

$$Y^{(k+1)} = A \times_1 U^{(k+1)^T} \times_3 W^{(k)^T}$$
$$V^{(k+1)} = \text{first } r_2 \text{ left singular vectors of } Y_{(2)}^{(k+1)}$$

$$Z^{(k+1)} = A \times_1 U^{(k+1)^T} \times_2 V^{(k+1)^T}$$
$$W^{(k+1)} = \text{first } r_3 \text{ left singular vectors of } Z_{(3)}^{(k+1)}$$

$$\Sigma^{(k+1)} = A \times_1 U^{(k+1)^T} \times_2 V^{(k+1)^T} \times_3 W^{(k+1)^T}$$

end

TUCKALS3 or HOOI: Kroonenberg and de Leeuw (1980), Kapteyn et al. (1986), De Lathauwer et al. (2000)

MATLAB ALS

```
% A = tensor
% r1,r2,r3 = desired ranks
for i = 1:maxits
  T = ttm(A,{V',W'},-1);
  [U,xx,zz] = svds(double(tensor_as_matrix(T,1)),r1);
  T = ttm(A,{U',W'},-2);
  [V,xx,zz] = svds(double(tensor_as_matrix(T,2)),r2);
  T = ttm(A,{U',V'},-3);
  [W,xx,zz] = svds(double(tensor_as_matrix(T,3)),r3);
end
S = ttm(A,{U',V',W'})
B = tucker_tensor(S,{U,V,W});
```

See `examples/hooi.m` for full method.

Sandia National Laboratories

Optimizing the Core

- Tucker decomposition is not uniquely determined
- Can freely multiply the components U,V,W by non-singular matrices R,S,T
  - This in turn changes the core Σ

$$\boxed{A = \Sigma \times_1 U \times_2 V \times_3 W}$$

Equivalent

$$\boxed{\begin{aligned} A &= \tilde{\Sigma} \times_1 (RU) \times_2 (SV) \times_3 (TW) \\ \tilde{\Sigma} &= \Sigma \times_1 R^{-1} \times_2 S^{-1} \times_3 T^{-1} \end{aligned}}$$

- Freedom can be used to modify Σ so that it has meaning
  - Maximize diagonal
  - Maximize variance (i.e., tends toward either 0 or 1 entries)
- See, e.g.,
  - Kiers (1992, 1997, 1998), Henrion (1993)

Sandia National Laboratories

Sparse Tensors in MATLAB

(time permitting)

Sandia National Laboratories

Storing Sparse N-Way Data

- Sparse Matrix Storage Schemes
  - For m x n matrix with q nonzeros
  - Compressed sparse row (CSR)
    - For each entry, store its column index and value
    - Store the start of each row plus the end of the last row
    - Storage = 2q + m + 1
    - Easy to pull out a row, hard to pull out a column
    - Can do analogous Compress Sparse Column (CSC)
    - Generally considered the most efficient storage format
  - Coordinate format
    - Store row and column index along with each value
    - Storage = 3q

- Sparse Multiway Data Storage Schemes
  - For m x n x p with q nonzeros
  - Compressed sparse mode
    - For each entry, stores its mode-n index and value
    - Store and end (per mode)
    - Storage = 2q + m(n+1) + 1
    - Various efficiencies are possible (e.g., skipping)
  - Store $A_{(3)}$ in CSR format
    - Storage = 2q + mn + 1
  - Store $A_{(3)}$ in CSC format
    - Storage = 2q + p + 1
  - Coordinate format
    - Store all mode indices with each value
    - Storage = 4q

Our choice

Sandia National Laboratories

Computing with Coordinate Format: $A \times_2 u$

- Let A be a 3rd order tensor of size m x n x p with q nonzeros
  - `Aidx` = indices stored as a q x 3 array
  - `Aval` = values stored as a q x 1 array
- Let u be an n-vector
  - `u` = n x 1 array
- How do we calculate $B = A \times_2 u$?
  - Without explicit for-loops!

```
[subs,junk,loc] =
   unique(Aidx(:,[1,3]),
   'rows');
vals = accumarray(loc,
   Avals .* u(Aidx(:,2)));
nz = find(vals);
Bsubs = subs(nz,:);
Bvals = vals(nz);
```

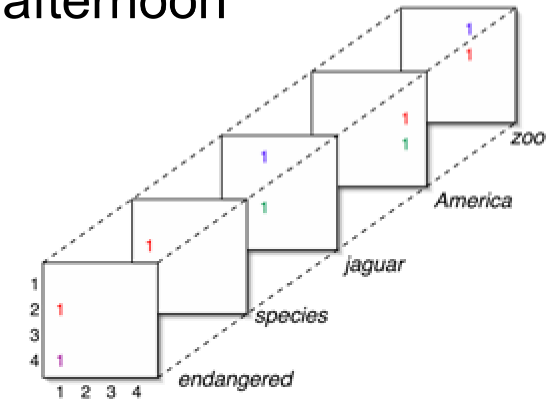
Sandia National Laboratories

Computational Efficiency

- In MATLAB, need to avoid explicitly looping over data
- Make extensive use of built-in MATLAB functions such as…
  - ismember
  - setdiff
  - unique
  - accumarray
  - cumprod / cumsum
  - sort / sortrows

Is anyone interested in a MATLAB toolbox for sparse tensors?

- Have been able to run a greedy PARAFAC and similar algorithms on problems of size
  - 50k x 50k x 70k
  - ½ million nonzeros
- See Brett's talk on Friday afternoon

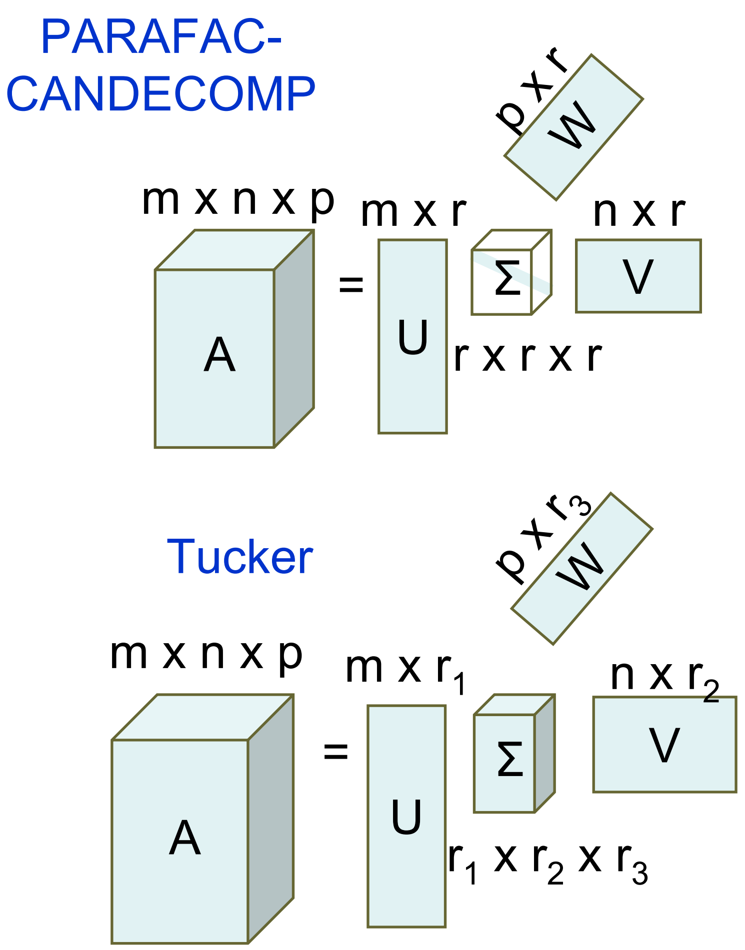


Sandia National Laboratories

Summary & References

Summary

- Part I
  - Basics of N-way arrays
  - Tensor decompositions
- Part II
  - Computing tensor decompositions
  - Choosing the core in the Tucker decomposition
  - Sparse tensors



Thanks for your attention.
For more information,
Tamara G. Kolda
tgkolda@sandia.gov

References

- C. A. Andersson and R. Bro. **The N-way toolbox for MATLAB**. *Chemometr. Intell. Lab.*, 52(1):1–4, 2000. See also http://www.models.kvl.dk/source/nwaytoolbox/.
- Brett W. Bader and Tamara G. Kolda. **MATLAB tensor classes for fast algorithm prototyping**. Technical Report SAND2004-5187, Sandia National Laboratories, Albuquerque, NM 87185 and Livermore, CA 94550, October 2004. Submitted to ACM Trans. Math. Software.
- J. D. Carroll and J. J. Chang. **Analysis of individual differences in multidimensional scaling via an N-way generalization of 'Eckart-Young' decomposition**. *Psychometrika*, 35:283–319, 1970.
- Lieven De Lathauwer, Bart De Moor, and Joos Vandewalle. **A multilinear singular value decomposition**. *SIAM J. Matrix Anal. A.*, 21(4):1253–1278, 2000.
- Lieven De Lathauwer, Bart De Moor, and Joos Vandewalle. **On the best rank-1 and rank-(R1, R2, dots , RN) approximation of higher-order tensors**. *SIAM J. Matrix Anal. A.*, 21(4):1324–1342, 2000.
- Richard A. Harshman. **Foundations of the PARAFAC procedure: models and conditions for an ``explanatory" multi-modal factor analysis**. *UCLA working papers in phonetics*, 16:1–84, 1970.
- Richard A. Harshman. **An index formulism that generalizes the capabilities of matrix notation and algebra to n-way arrays**. *J. Chemometr.*, 15(9):689–714, August 2001. (doi:10.1002/cem.665)
- René Henrion. **Body diagonalization of core matrices in three-way principal components analysis: Theoretical bounds and simulation**. *J. Chemometr.*, 7(6):477–494, 1993. (doi:10.1002/cem.1180070604)
- René Henrion. **N-way principal component analysis theory, algorithms and applications**. *Chemometr. Intell. Lab.*, 25(1):1–23, September 1994.
- Henk A.L. Kiers. **Joint orthomax rotation of the core and component matrices resulting from three-mode principal components analysis**. *J. Classification*, 15(2):245 – 263, February 1998. (doi:10.1007/s003579900033)
- Henk A. L. Kiers. **Towards a standardized notation and terminology in multiway analysis**. *J. Chemometr.*, 14:105–122, 2000.
- Pieter M. Kroonenberg and Jan De Leeuw. **Principal component analysis of three-mode data by means of alternating least squares algorithms**. *Psychometrika*, 45(1):69–97, March 1980.
- Joseph B. Kruskal. **Three-way arrays: rank and uniqueness of trilinear decompositions, with application to arithmetic complexity and statistics**. *Linear Algebra Appl.*, 18(2):95–138, 1977.
- Ledyard R. Tucker. **Some mathematical notes on three-mode factor analysis**. *Psychometrika*, 31:279–311, 1966.

Sandia National Laboratories