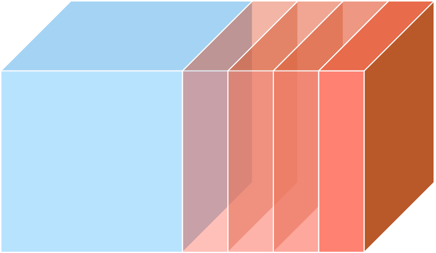
# Updating tensor decompositions

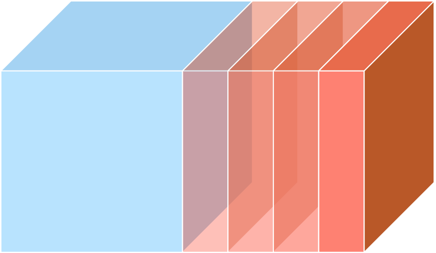
**Michiel Vandecappelle**
**michiel.vandecappelle@kuleuven.be**
Nico Vervliet
Martijn Boussé
Lieven De Lathauwer

EURASIP Summer School on
Tensor-Based Signal Processing
August 29, 2018

# Tensors can change over time



# Tensors can change over time

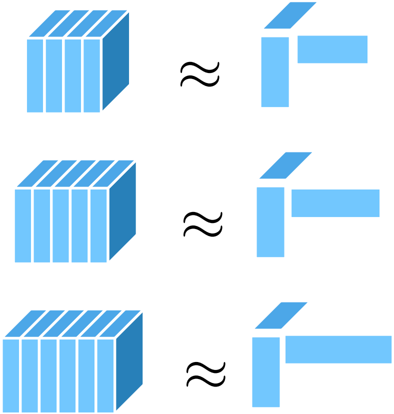


- Data only becomes available gradually
- The data is non-stationary
- The full tensor does not fit into memory

# Tensors can change over time

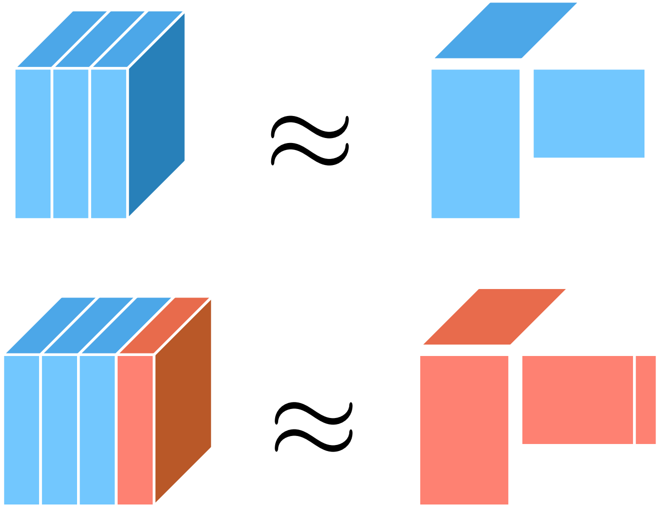
- Example: EEG data
  - New EEG samples are obtained
  - Old data becomes outdated
  - Test subjects are added/removed
  - Mobile EEG

# Recomputing the full decomposition can be infeasible or unnecessary



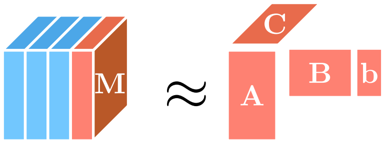
- The new data arrives too fast
- The full tensor takes up too much memory
- We want to locally improve the accuracy of the decomposition

# Can tensor decompositions adapt to changes in the tensor?



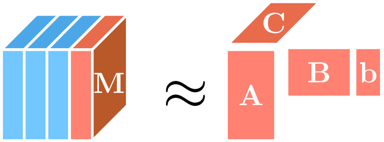
# CPD updating

- A new slice $\mathbf{M}$ gets added to the tensor



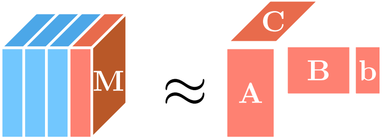
# CPD updating

- A new slice $\mathbf{M}$ gets added to the tensor
- Factor matrix $\mathbf{B}$ obtains an extra column, $\mathbf{b}$



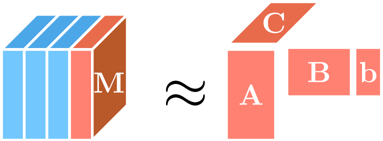
# CPD updating

- A new slice $\mathbf{M}$ gets added to the tensor
- Factor matrix $\mathbf{B}$ obtains an extra column, $\mathbf{b}$
- The original factor matrices $\mathbf{A}$, $\mathbf{B}$ and $\mathbf{C}$ are updated



# CPD updating

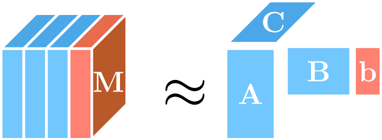
- A new slice $\mathbf{M}$ gets added to the tensor
- Factor matrix $\mathbf{B}$ obtains an extra column, $\mathbf{b}$
- The original factor matrices $\mathbf{A}$, $\mathbf{B}$ and $\mathbf{C}$ are updated
- The algorithm should be fast, accurate and memory-efficient



# Existing algorithms are sequential

- Estimate $\mathbf{b}$ with its least squares solution:

$$\left[(\mathbf{C}_{\mathsf{old}}^{\mathsf{T}}\mathbf{C}_{\mathsf{old}}) * (\mathbf{A}_{\mathsf{old}}^{\mathsf{T}}\mathbf{A}_{\mathsf{old}})\right]^{-1}(\mathbf{C}_{\mathsf{old}} \odot \mathbf{A}_{\mathsf{old}})^{\mathsf{T}}\mathsf{vec}(\mathbf{M})$$

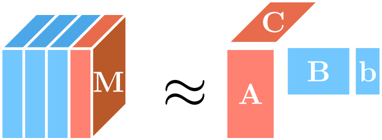


# Existing algorithms are sequential

- Estimate $\mathbf{b}$ with its least squares solution:

$$\left[(\mathbf{C}_{\mathsf{old}}^{\mathsf{T}}\mathbf{C}_{\mathsf{old}}) * (\mathbf{A}_{\mathsf{old}}^{\mathsf{T}}\mathbf{A}_{\mathsf{old}})\right]^{-1}(\mathbf{C}_{\mathsf{old}} \odot \mathbf{A}_{\mathsf{old}})^{\mathsf{T}}\mathsf{vec}(\mathbf{M})$$

- Update $\mathbf{A}$ and $\mathbf{C}$



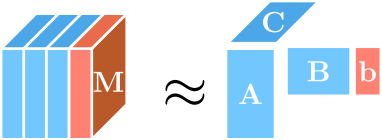
# Existing algorithms are sequential

- Estimate $\mathbf{b}$ with its least squares solution:

$$\left[(\mathbf{C}_{\text{old}}^{\mathsf{T}}\mathbf{C}_{\text{old}}) * (\mathbf{A}_{\text{old}}^{\mathsf{T}}\mathbf{A}_{\text{old}})\right]^{-1}(\mathbf{C}_{\text{old}} \odot \mathbf{A}_{\text{old}})^{\mathsf{T}}\text{vec}(\mathbf{M})$$

- Update $\mathbf{A}$ and $\mathbf{C}$
- Refine $\mathbf{b}$:

$$\left[(\mathbf{C}_{\text{new}}^{\mathsf{T}}\mathbf{C}_{\text{new}}) * (\mathbf{A}_{\text{new}}^{\mathsf{T}}\mathbf{A}_{\text{new}})\right]^{-1}(\mathbf{C}_{\text{new}} \odot \mathbf{A}_{\text{new}})^{\mathsf{T}}\text{vec}(\mathbf{M})$$



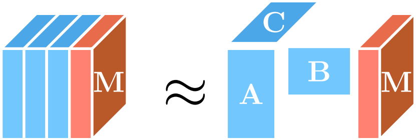
# All-at-once CPD updating

Adapt batch NLS CPD algorithm to the updating context

# All-at-once CPD updating

Adapt batch NLS CPD algorithm to the updating context

- Replace old slices by an approximation (CPD, MLSVD . . . )



# All-at-once CPD updating

Adapt batch NLS CPD algorithm to the updating context

- Replace old slices by an approximation (CPD, MLSVD . . . )
- Derive efficient expressions for objective function, gradient and Hessian approximation $\mathbf{J}^\mathsf{T}\mathbf{J}$ by exploiting structure of old slices
  - Split the computation in parts depending on the old data and parts depending on new slices
  - Use the structured tensor framework for the old data

# All-at-once CPD updating

Adapt batch NLS CPD algorithm to the updating context

- Replace old slices by an approximation (CPD, MLSVD . . . )
- Derive efficient expressions for objective function, gradient and Hessian approximation $\mathbf{J}^\mathsf{T}\mathbf{J}$ by exploiting structure of old slices
- Apply Conjugate Gradients to solve the system $\mathbf{J}^\mathsf{T}\mathbf{J}\mathbf{p} = -\mathbf{g}$
  - Forming and inverting $\mathbf{J}^\mathsf{T}\mathbf{J}$ is not needed
  - Products of the form $\mathbf{J}^\mathsf{T}\mathbf{J}\mathbf{x}$ can be computed efficiently

# All-at-once CPD updating

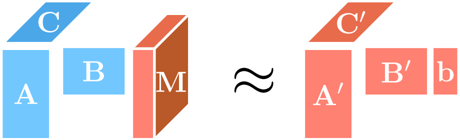
Adapt batch NLS CPD algorithm to the updating context

- Replace old slices by an approximation (CPD, MLSVD ...)
- Derive efficient expressions for objective function, gradient and Hessian approximation $\mathbf{J}^\mathsf{T}\mathbf{J}$ by exploiting structure of old slices
- Apply Conjugate Gradients to solve the system $\mathbf{J}^\mathsf{T}\mathbf{J}\mathbf{p} = -\mathbf{g}$
- Use block-Jacobi preconditioner $\mathbf{M}$
  - Left-multiply both sides of $\mathbf{J}^\mathsf{T}\mathbf{J}\mathbf{p} = -\mathbf{g}$ by $\mathbf{M}^{-1}$
  - Good choice of $\mathbf{M}$ improves convergence of CG algorithm

# All-at-once CPD updating

Adapt batch NLS CPD algorithm to the updating context

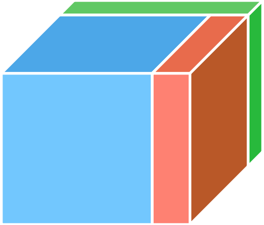
- Replace old slices by an approximation (CPD, MLSVD . . . )
- Derive efficient expressions for objective function, gradient and Hessian approximation $\mathbf{J}^\mathsf{T}\mathbf{J}$ by exploiting structure of old slices
- Apply Conjugate Gradients to solve the system $\mathbf{J}^\mathsf{T}\mathbf{J}\mathbf{p} = -\mathbf{g}$
- Use block-Jacobi preconditioner $\mathbf{M}$
- Limit number of NLS and CG iterations



# Numerous generalizations of the algorithm exist

# Numerous generalizations of the algorithm exist

- Applicable to $(N-1)$-th order updates of $N$th-order tensors in any number of modes simultaneously



# Numerous generalizations of the algorithm exist

- Applicable to $(N-1)$-th order updates of $N$th-order tensors in any number of modes simultaneously
- Truncated exponential windows are easily applied

# Numerous generalizations of the algorithm exist

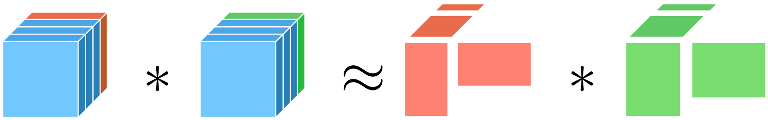
- Applicable to $(N-1)$-th order updates of $N$th-order tensors in any number of modes simultaneously
- Truncated exponential windows are easily applied
- Rank changes are possible

# Numerous generalizations of the algorithm exist

- Applicable to $(N-1)$-th order updates of $N$th-order tensors in any number of modes simultaneously
- Truncated exponential windows are easily applied
- Rank changes are possible
- Extendable to structured or coupled CPDs

# Numerous generalizations of the algorithm exist

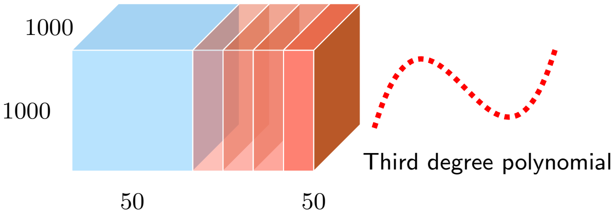
- Applicable to $(N-1)$-th order updates of $N$th-order tensors in any number of modes simultaneously
- Truncated exponential windows are easily applied
- Rank changes are possible
- Extendable to structured or coupled CPDs
- Weighted least squares (WLS) CPD updating
  - CPD of weight tensor is updated
  - This CPD is used in WLS update of the data tensor CPD



# Updating is both accurate and fast

# Updating is both accurate and fast

- Model with slowly evolving second mode and $20$dB SNR

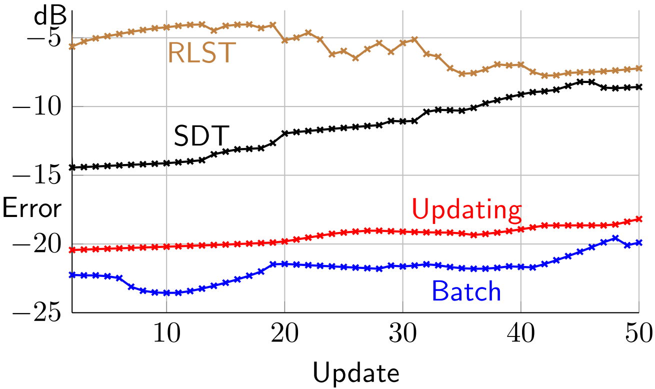


# Updating is both accurate and fast

- Model with slowly evolving second mode and $20$dB SNR
- Comparison of batch and updating methods
  - Batch CPD-NLS and CPD-ALS (Sorber et al. 2013)
  - PARAFAC-SDT and PARAFAC-RLST (Nion et al. 2009)
  - CPD updating

# Updating is both accurate and fast

- Model with slowly evolving second mode and $20$dB SNR
- Comparison of batch and updating methods
- Mean errors of the CPD approximation



# Updating is both accurate and fast

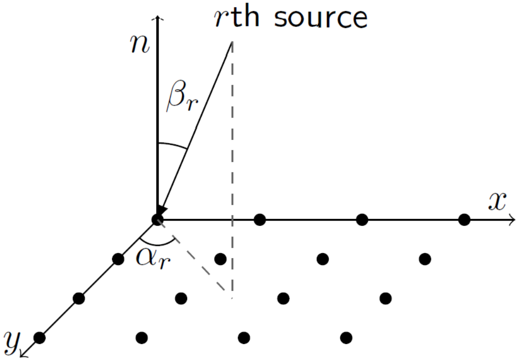
- Model with slowly evolving second mode and $20$dB SNR
- Comparison of batch and updating methods
- Mean errors of the CPD approximation
- CPU-time of the CPD approximation (ms)

| $R$ | $2$ | $3$ | $4$ | $5$ | $6$ |
|---|---|---|---|---|---|
| Updating | 60 | 81 | 104 | 140 | 169 |
| NLS | 2375 | 4464 | 2557 | 3563 | 5522 |
| ALS | 910 | 1222 | 1400 | 1401 | 2352 |
| SDT | 48 | 71 | 98 | 136 | 172 |
| RLST | 570 | 607 | 623 | 775 | 822 |

# NLS updating enables real-time CPD applications

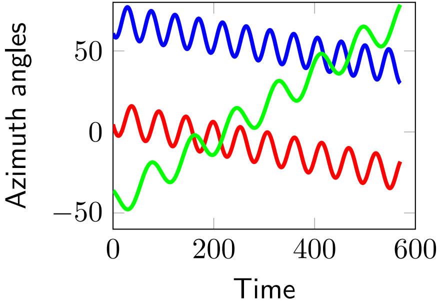
# NLS updating enables real-time CPD applications

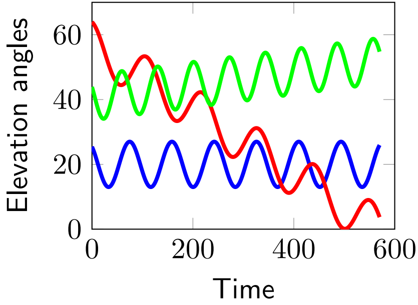
- Direction-of-arrival estimation using a uniform rectangular array (URA)



# NLS updating enables real-time CPD applications

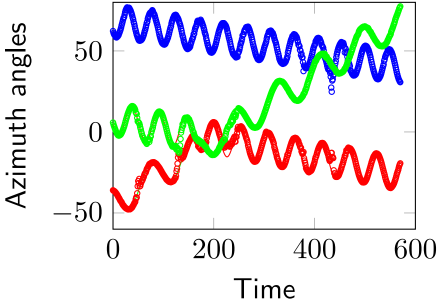
- Direction-of-arrival estimation using a uniform rectangular array (URA)
- Three moving sources, SNR 10dB

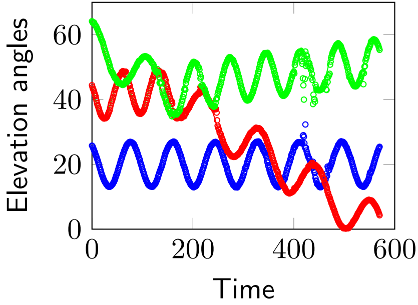


# NLS updating enables real-time CPD applications

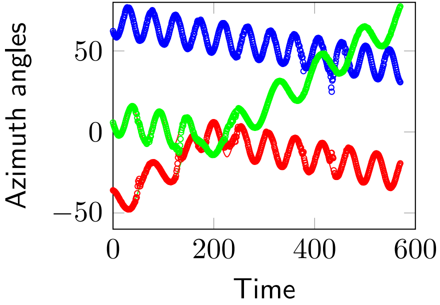
- Direction-of-arrival estimation using a uniform rectangular array (URA)
- Three moving sources, SNR 10dB
- Azimuth and elevation angles of sources are recovered from CPD of data tensor

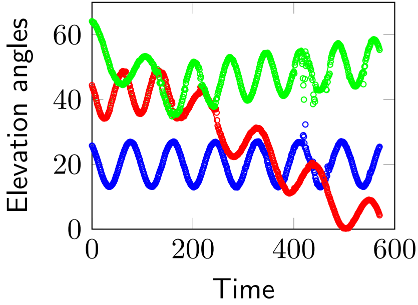


# NLS updating enables real-time CPD applications

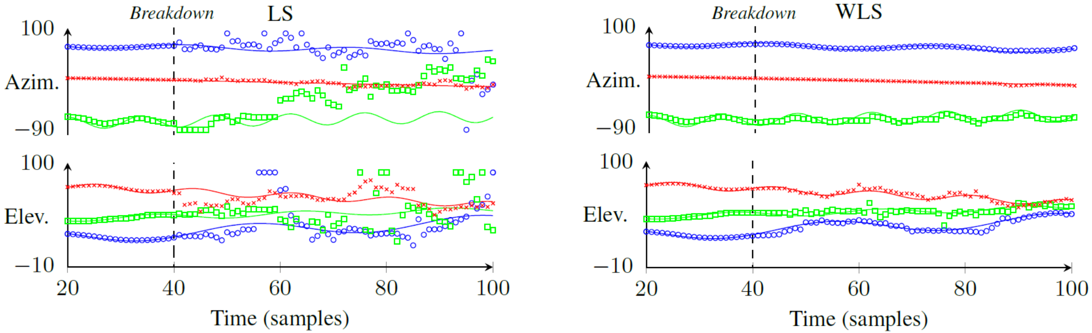
- Direction-of-arrival estimation using a uniform rectangular array (URA)
- Three moving sources, SNR 10dB
- Azimuth and elevation angles of sources are recovered from CPD of data tensor
- NLS updating: 6-8ms per update



# NLS updating enables real-time CPD applications

- WLS NLS CPD updating
  - Direction-of-arrival estimation with three moving sources
  - Some sensors break down, leading to bad readings
  - Less weight is given to readings of these sensors



# Main advantages of all-at once CPD updating

# Main advantages of all-at once CPD updating

- Fast and accurate

# Main advantages of all-at once CPD updating

- Fast and accurate
- Versatile

# Main advantages of all-at once CPD updating

- Fast and accurate
- Versatile
- Low memory requirements $\mathcal{O}(\text{New slice})$

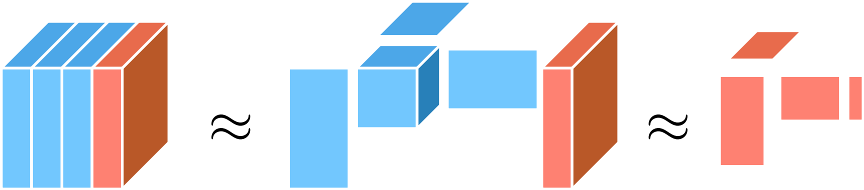
# Improvements can still be made

# Improvements can still be made

- Automatic rank-adaptation

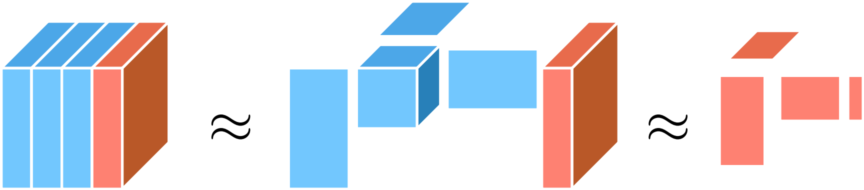
# Improvements can still be made

- Automatic rank-adaptation
- Additionally track an MLSVD or higher-rank CPD of the tensor



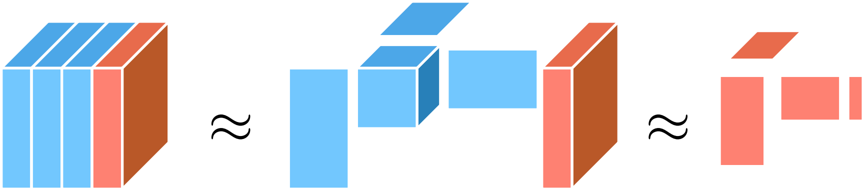
# Improvements can still be made

- Automatic rank-adaptation
- Additionally track an MLSVD or higher-rank CPD of the tensor
  - Compromise between storing full tensor and only CPD



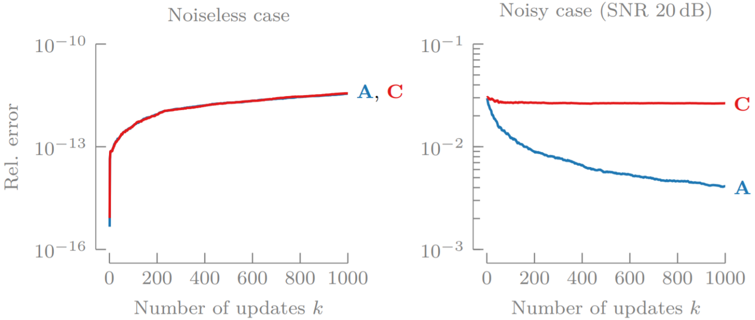
# Improvements can still be made

- Automatic rank-adaptation
- Additionally track an MLSVD or higher-rank CPD of the tensor
  - Compromise between storing full tensor and only CPD
  - Track enough data for rank-increases



# Improvements can still be made

- Automatic rank-adaptation
- Additionally track an MLSVD or higher-rank CPD of the tensor
  - Compromise between storing full tensor and only CPD
  - Track enough data for rank-increases
  - Countering numerical error accumulation
    - Numerical error increases for all modes
    - Statistical error decreases for non-updated modes



# Further possibilities for updating algorithms

- Other decompositions (MLSVD, BTD)
- Different cost functions
- Tensor data changing in other ways

# Updating tensor decompositions

**Michiel Vandecappelle**
**michiel.vandecappelle@kuleuven.be**
Nico Vervliet
Martijn Boussé
Lieven De Lathauwer

EURASIP Summer School on
Tensor-Based Signal Processing
August 29, 2018