Distributed spatial filtering and feature fusion

Alexander Bertrand, Cem Ates Musluoglu and Charles Hovine

Download slides: https://tinyurl.com/EUSIPCO23

EUSIPCO2023
31ST EUROPEAN SIGNAL PROCESSING CONFERENCE
SEPTEMBER 4–8, 2023 • HELSINKI, FINLAND

KU LEUVEN
STADIUS Center for Dynamical Systems, Signal Processing and Data Analytics

erc
European Research Council
Established by the European Commission

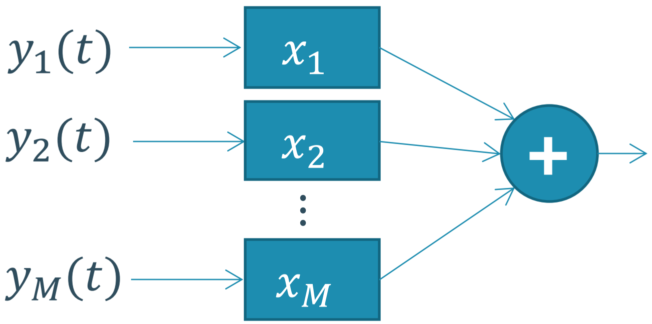
Outline

I- Spatial filtering and distributed signal processing

Preliminaries

Spatial Filtering

- **Spatial filter**: linearly combine channels of a multi-channel signal $\mathbf{y}(t) = [y_1(t) \;\; y_2(t) \;\; \ldots \;\; y_M(t)]^T$
  - **MISO**: $z(t) = \mathbf{x}^T \mathbf{y}(t)$ with $\mathbf{x} \in R^M$ → M input channels, 1 output channel



$$z(t) = \sum_m x_m y_m(t)$$

  - **MIMO**: $\mathbf{z}(t) = X^T \mathbf{y}(t)$ with $X \in R^{M \times Q}$ → M input channels, Q output channels

Data-Driven Spatial Filtering: generic representation (*)

- **Data-driven**: optimize X as a function of the sensor data (statistics)

$$\begin{aligned}\underset{X}{\text{minimize}}\ & f(X^T\mathbf{y}(t))\\ \text{subject to}\ & g_j(X^T\mathbf{y}(t)) = 0,\\ & h_j(X^T\mathbf{y}(t)) \leq 0\end{aligned}$$

→ We optimize X, but are generally (more) interested in $\boldsymbol{z}(t) = X^T\boldsymbol{y}(t)$.

Example: minimum mean squared error (MMSE), a.k.a. multi-channel Wiener filter (MWF)

$$\min\ \mathbb{E}[\|\mathbf{d}(t) - X^T\mathbf{y}(t)\|_F^2]$$

Example: linearly constrained minimum variance (LCMV) beamformer

$$\begin{aligned}&\min\ \mathbb{E}[\|X^T\mathbf{y}(t)\|_F^2]\\ &\text{s.t. } X^T B = H\end{aligned}$$

Example: max-SNR filtering

$$\max\ \frac{\mathbb{E}[\|X^T\mathbf{y}(t)\|_F^2]}{\mathbb{E}[\|X^T\mathbf{n}(t)\|_F^2]}$$

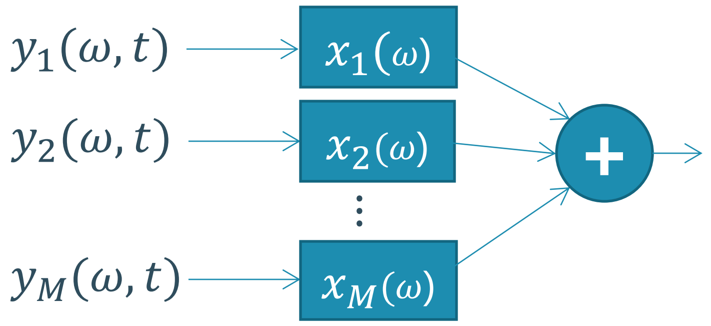
Example: principal component analysis (PCA)

$$\begin{aligned}&\max\ \mathbb{E}[\|X^T\mathbf{y}(t)\|_F^2]\\ &\text{s.t. } X^T X = I\end{aligned}$$

(*) Will assume real-valued signals for simplicity, all results can be extended to complex-valued signals

Data-driven Spatial Filtering

- Typical assumption: $\boldsymbol{y}(t)$ is ergodic and (short-term) stationary
- If statistics of $\boldsymbol{y}(t)$change (slowly) over time → **adaptive** spatial filter
- Extensions to convolutive spatio-temporal filter via time domain or Short-Time Fourier Transform domain (abstracted here)



$$z(\omega, t) = \sum_m x_m(\omega) y_m(\omega, t)$$

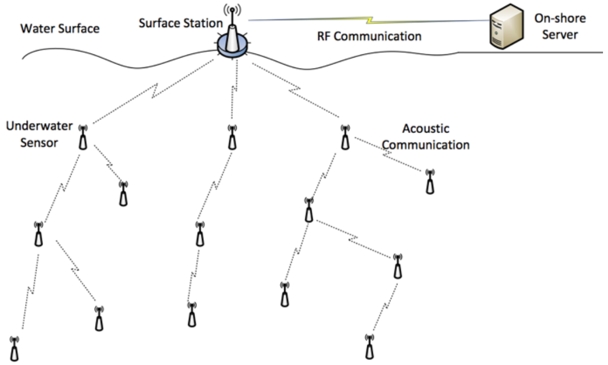
Wireless Sensor Networks

Wireless Sensor Network:

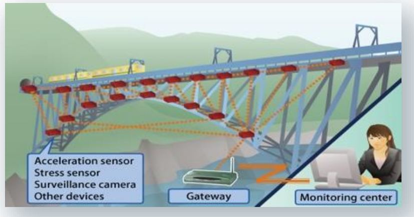
A network of inter-connected devices equipped with **sensing**, **computing** and **communication** capabilities.

Camera networks

Under-water sensing

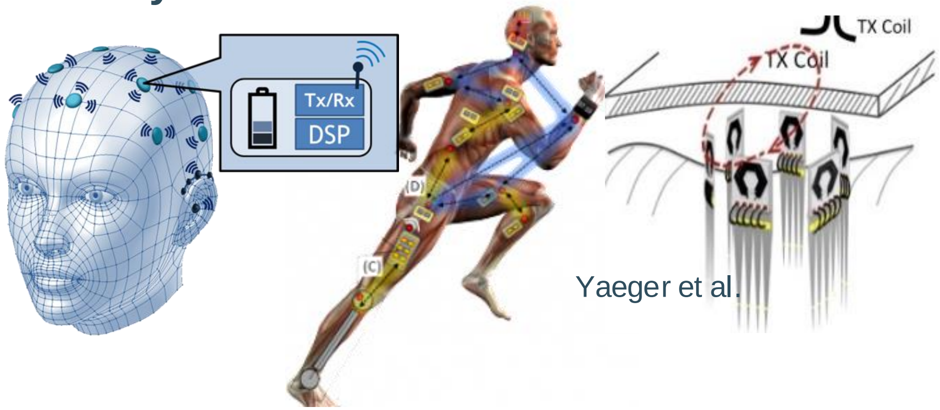


structural health monitoring



Microphone networks
(a.k.a. acoustic sensor networks)

Physiological sensor networks & body-area networks



Yaeger et al.

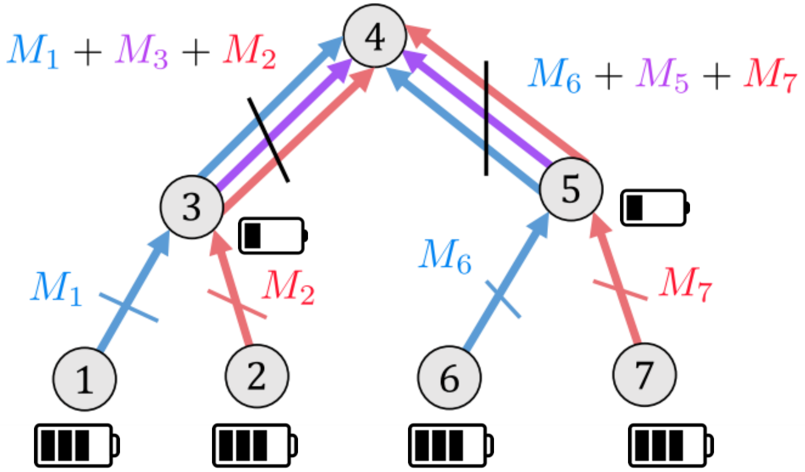
Smart homes & Ambient assisted living

And more…
(radar, environmental monitoring, …)

Distributed Signal Processing
Motivation

Data centralization in a fusion center:

- Simple, use of off-the-shelf SP algorithms
- Wireless transmission of raw sensor data (impossible?)
- Large computational requirement at the fusion center
- Single point of failure

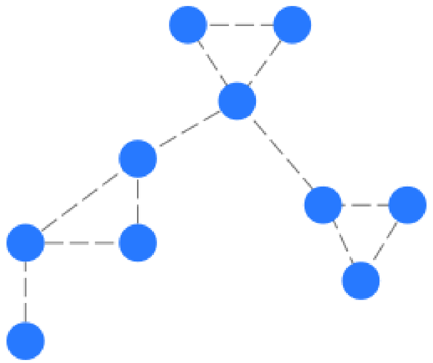


Distributed Signal Processing
Motivation

Distributed processing:

No Fusion Center – All nodes equal

- Scalable / compressive data aggregation
- Computational burden is shared
- Robust against node failure
- SP algorithm design is tedious

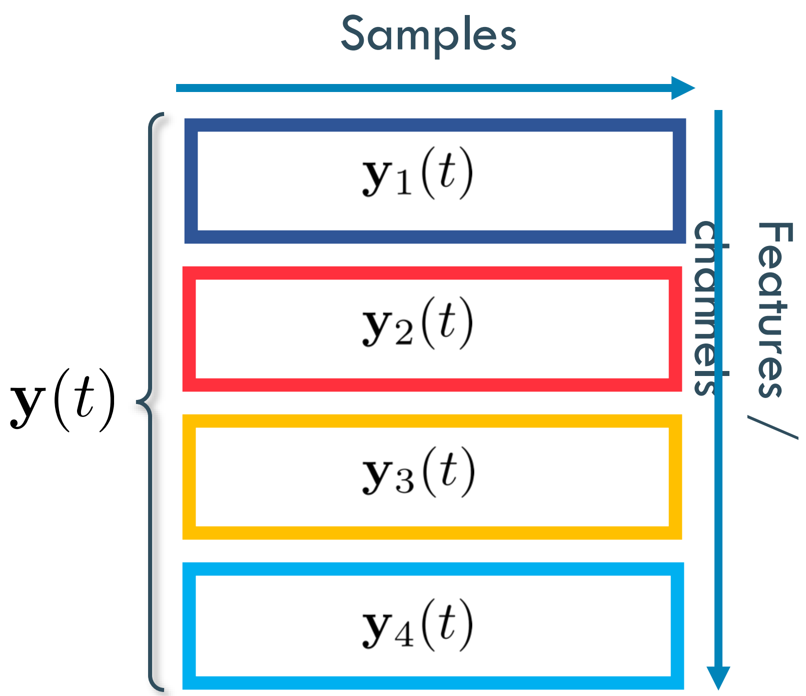


Distributed Signal Processing

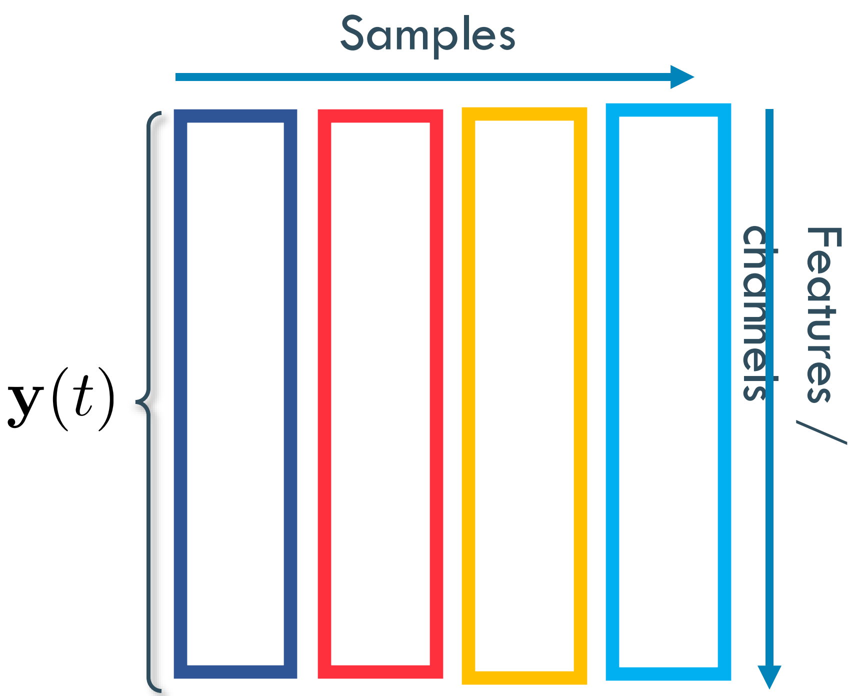
Problem Taxonomy

Feature-split

(spatial filtering: features = channels)



Sample-split

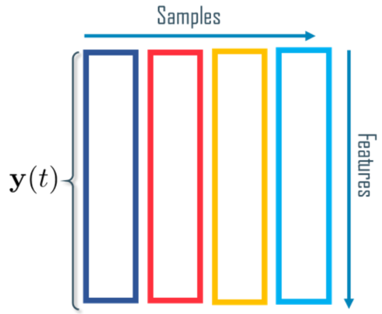


Distributed Signal Processing
Problem Taxonomy

- **Sample-split** typically leads to a **node-separable objective** with a **joint variable** X

$$\min_X \sum_k f_k(X)$$

!! f_k only depends on data from node k !!



- = focus of most distributed SP ‘workhorse’ algorithms

(consensus, diffusion, ADMM, PDMM, gossip, …)

- Typical strategy:

$$\min_X \sum_k f_k(X_k)$$

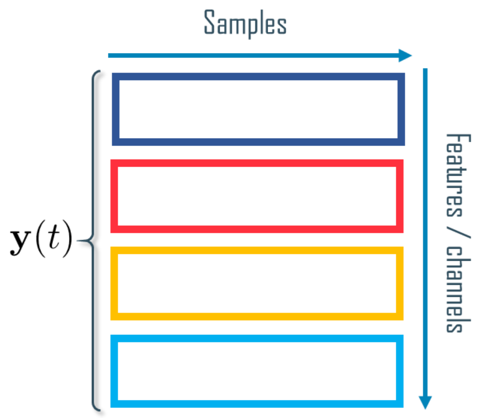
$$\text{s.t. } X_i = X_j \ \forall\, i, j$$

Only communicate intermediate estimates X_k^i (usually no sensor data)

Distributed Signal Processing
Main challenge

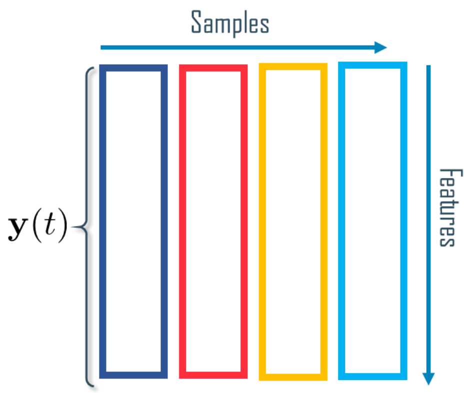
Objective is *not* node-separable

$$\min_X f(X^T \mathbf{y}(t))$$



node-separable objective

$$\min_X \sum_k f_k(X)$$

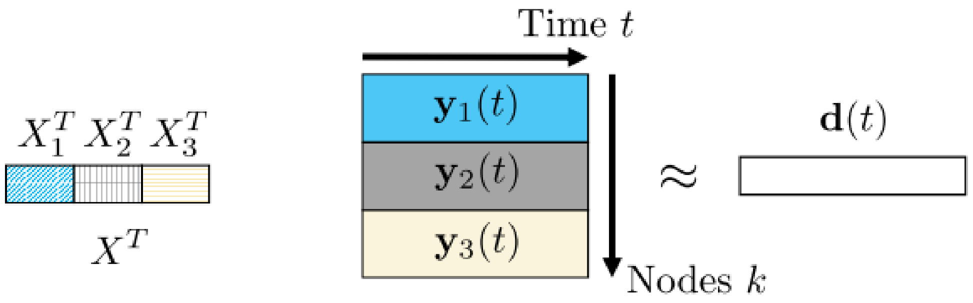


Distributed Signal Processing
Main challenge

Node-separable argument

MMSE example

$$\sum_t ||\mathbf{d}(t) - \sum_k X_k^T \mathbf{y}_k(t)||^2$$

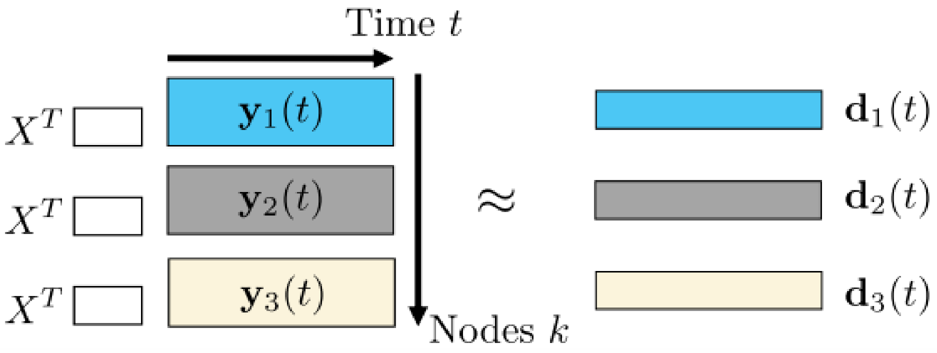


Interest is (usually) in output <u>signal</u> $X^T\boldsymbol{y}(t)$

Node-separable objective

MMSE example

$$\sum_k \sum_t ||\mathbf{d}_k(t) - X^T \mathbf{y}_k(t)||^2$$



Interest is (usually) in estimated <u>parameter</u> $\hat{X}$

Distributed Signal Processing
Main challenge

Feature / channel split: how to deal with **inter-node signal covariance**?

$$E\left[\|X^T\mathbf{y}(t)\|_F^2\right] = \mathrm{tr}(X^T E\left[\mathbf{y}(t)\mathbf{y}^T(t)\right] X) = \mathrm{tr}(X^T R_{\mathbf{yy}} X)$$

$$E[\mathbf{y}\mathbf{y}^T] = R_{\mathbf{yy}} = \begin{bmatrix} R_{11} & R_{12} & R_{13} & R_{14} \\ R_{21} & R_{22} & R_{23} & R_{24} \\ R_{31} & R_{32} & R_{33} & R_{34} \\ R_{41} & R_{42} & R_{43} & R_{44} \end{bmatrix}$$

Inter-node covariance

Node-specific covariance

Distributed Signal Processing

Main challenge

Feature-split

(spatial filtering: features = channels)

$$R_{\mathbf{yy}} = \begin{bmatrix} R_{11} & R_{12} & R_{13} & R_{14} \\ R_{21} & R_{22} & R_{23} & R_{24} \\ R_{31} & R_{32} & R_{33} & R_{34} \\ R_{41} & R_{42} & R_{43} & R_{44} \end{bmatrix}$$

None of the nodes observes the network-wide covariance matrix R_{yy}

Sample-split

$$\begin{bmatrix} S_{11} & S_{12} & S_{13} & S_{14} \\ S_{21} & S_{22} & S_{23} & S_{24} \\ S_{31} & S_{32} & S_{33} & S_{34} \\ S_{41} & S_{42} & S_{43} & S_{44} \end{bmatrix} \quad \begin{bmatrix} S_{11} & S_{12} & S_{13} & S_{14} \\ S_{21} & S_{22} & S_{23} & S_{24} \\ S_{31} & S_{32} & S_{33} & S_{34} \\ S_{41} & S_{42} & S_{43} & S_{44} \end{bmatrix}$$

$$+$$

$$\begin{bmatrix} S_{11} & S_{12} & S_{13} & S_{14} \\ S_{21} & S_{22} & S_{23} & S_{24} \\ S_{31} & S_{32} & S_{33} & S_{34} \\ S_{41} & S_{42} & S_{43} & S_{44} \end{bmatrix} \quad \begin{bmatrix} S_{11} & S_{12} & S_{13} & S_{14} \\ S_{21} & S_{22} & S_{23} & S_{24} \\ S_{31} & S_{32} & S_{33} & S_{34} \\ S_{41} & S_{42} & S_{43} & S_{44} \end{bmatrix}$$

- *Each node* observes the **full covariance matrix** R_{yy}
- Aggregated sample covariance can be obtained with an in-network sum

Distributed Spatial filtering → *ad hoc* treatment in literature

, Hovine C.

Distributed Spatial filtering

Common (ad hoc) strategies

#1: Use centralized algorithm (e.g. stochastic gradient descent) and compute each network-wide inner product via some type of **consensus iteration**

Example: Oja's algorithm for PCA

$$\mathbf{x}^{i+1} = \mathbf{x}^i + \mu \left(\beta^i \mathbf{y}(i) - \left(\beta^i\right)^2 \mathbf{x}^i \right) \text{ with } \beta^i = \mathbf{y}(i)^T \mathbf{x}^i = \sum_k y_k(i) x_k^i$$

Oja's iterative procedure
(outer loop iteration)

Consensus iteration for in-network
averaging / summation (nested iteration)

Distributed Spatial filtering

Common (ad hoc) strategies

#1: Use centralized algorithm (e.g. stochastic gradient descent) and compute each network-wide inner product via some type of **consensus iteration**

1) **communication-intensive**:

   nested iterations (2 time scales)

   Multiple transmissions for each *single* collected sample

2) **does not scale** well with network size

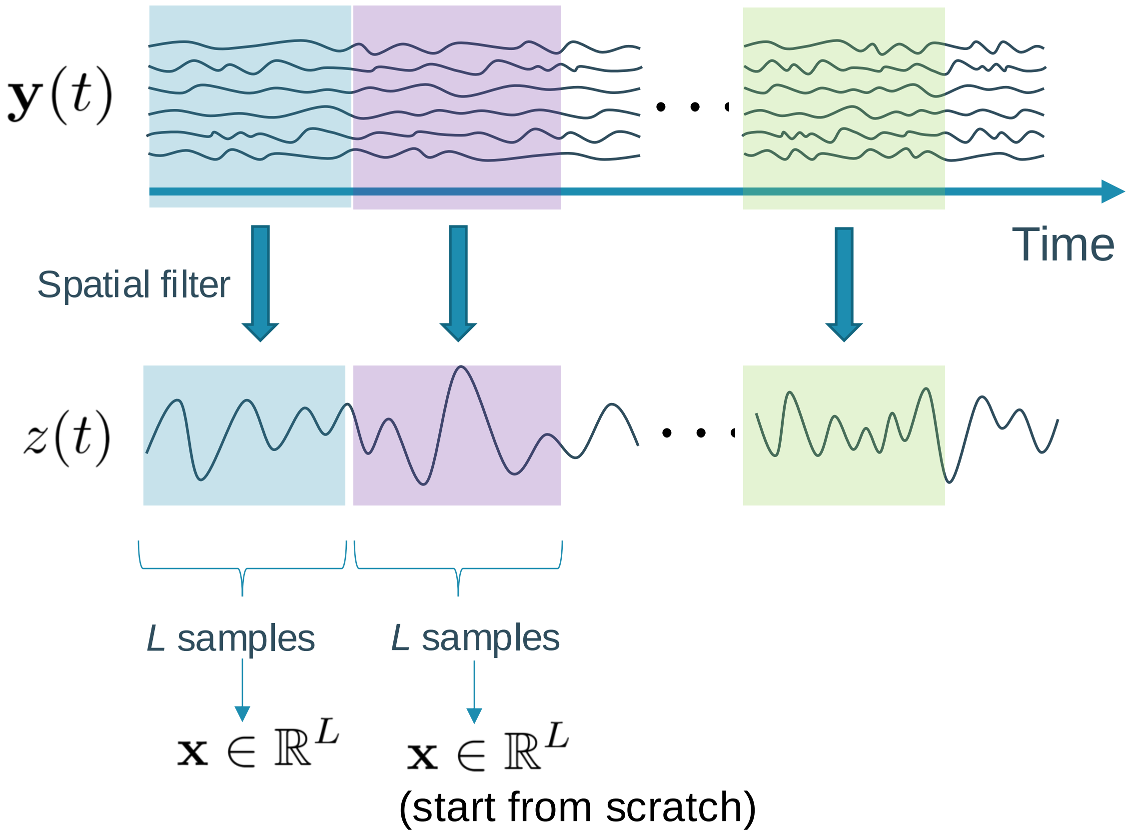
3) strategy not always applicable (depends on chosen centralized algo)

4) often **per-sample transmissions** $(i = t)$ → inefficient compared to sample-block transmissions

Distributed Spatial filtering

Common (ad hoc) strategies

#2: Treat block of samples of the filter output signal as consensus variable $\boldsymbol{x}$ and run a workhorse distributed algo (consensus, diffusion, ADMM, …) on each separate block



Distributed Spatial filtering

Common (ad hoc) strategies

#2: Treat block of samples of the filter output signal as consensus variable $\boldsymbol{x}$ and run a workhorse distributed algo (consensus, diffusion, ADMM, …) on each separate block

1) **communication-intensive :**

   - Solving 1 block of L samples = multiple iterations, each one requiring transmission of L-dimensional vector

   - Start from scratch for the next block of L samples

2) **does not scale** well with network size

3) strategy not always applicable (depends on optimization problem)

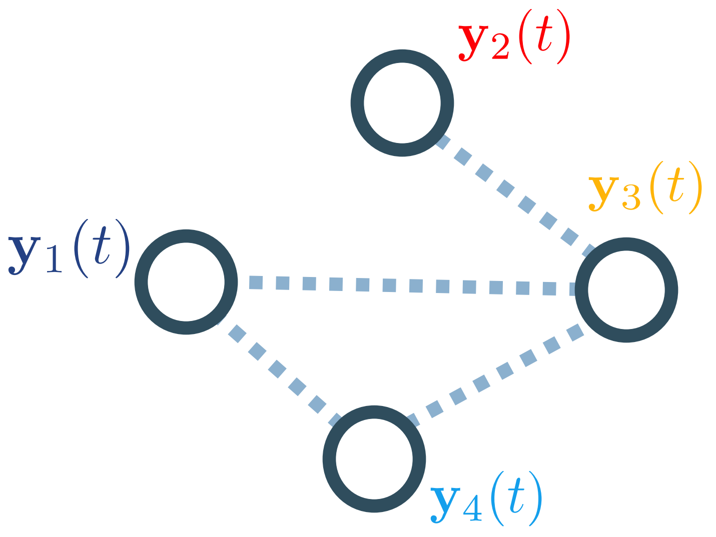
Distributed Spatial filtering

Common (ad hoc) strategies

#3: Make (unrealistic) assumptions on covariance matrix, e.g.

- $\boldsymbol{y}(t)$ or noise is spatially white
- $R_{\boldsymbol{yy}}$ has same structure as the communication network

$$R_{\mathbf{yy}} = \begin{bmatrix} R_{11} & 0 & R_{13} & R_{14} \\ 0 & R_{22} & R_{23} & 0 \\ R_{31} & R_{32} & R_{33} & R_{34} \\ R_{41} & 0 & R_{43} & R_{44} \end{bmatrix}$$



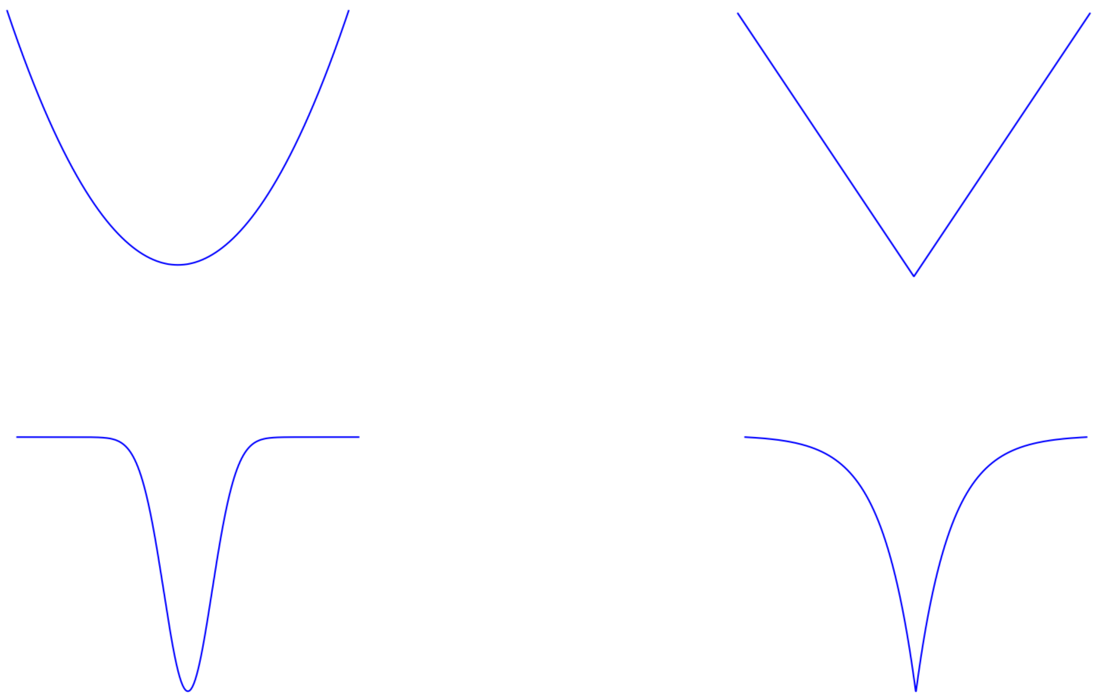
Distributed Signal Processing

Problem Taxonomy

Smooth | **Non-smooth**

Convex

Non-convex



Note: many spatial filtering problems are non-convex (PCA, max-SNR, CCA, …)

Distributed Signal Processing

Algorithms Overview

Algorithm	Exchanged data	Data split	Non-convex problems	Non-smooth problems	Adaptive
Consensus	Opt. variables	Samples	depends	depends	
Diffusion	Opt. variables	Samples			
ADMM or PDMM	Opt. variables	Samples (or Features/channels)			

Problem for spatial filtering
→ requires split in features/channels

Distributed Signal Processing

Algorithms Overview

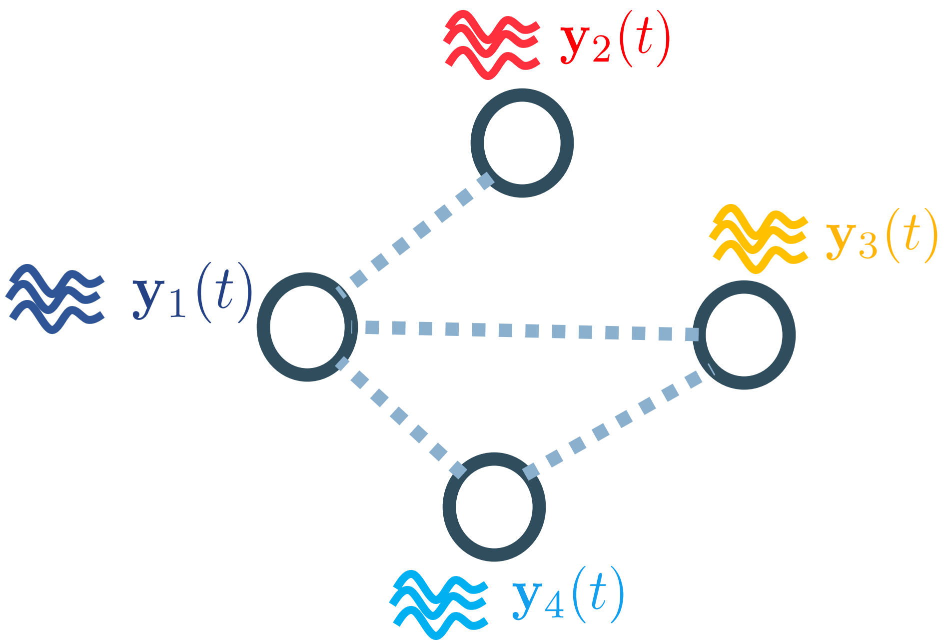
Algorithm	Exchanged data	Data split	Non-convex problems	Non-smooth problems	Adaptive
Consensus	Opt. variables	Samples	depends	depends	
Diffusion	Opt. variables	Samples			
ADMM or PDMM	Opt. variables	Samples (or Features/channels)			
DASF	Signals	Features/channels			

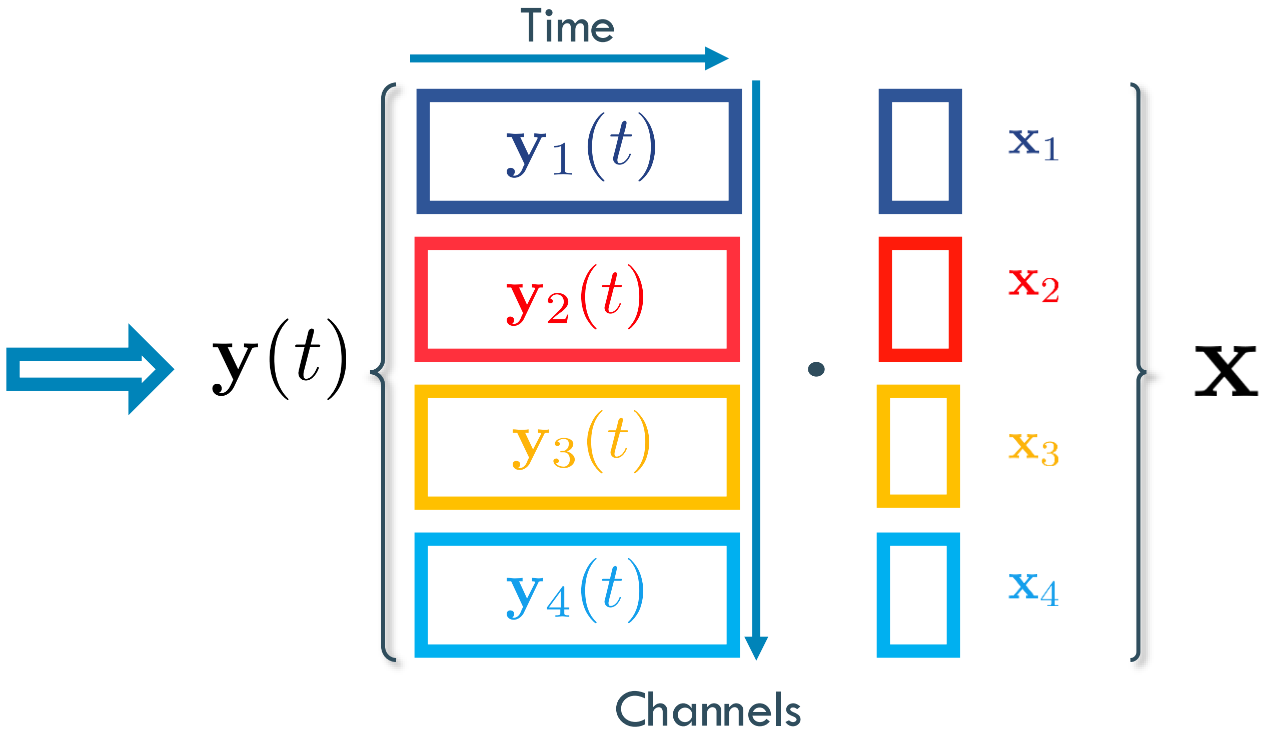
Spoiler Alert !

II- Towards a generic meta-algorithm for distributed data-driven spatial filtering

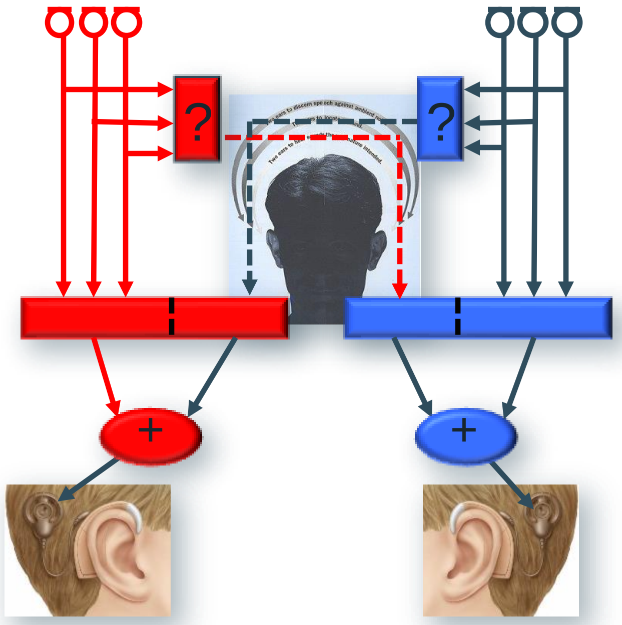
II.A- Preliminaries

Distributed spatial filtering setting

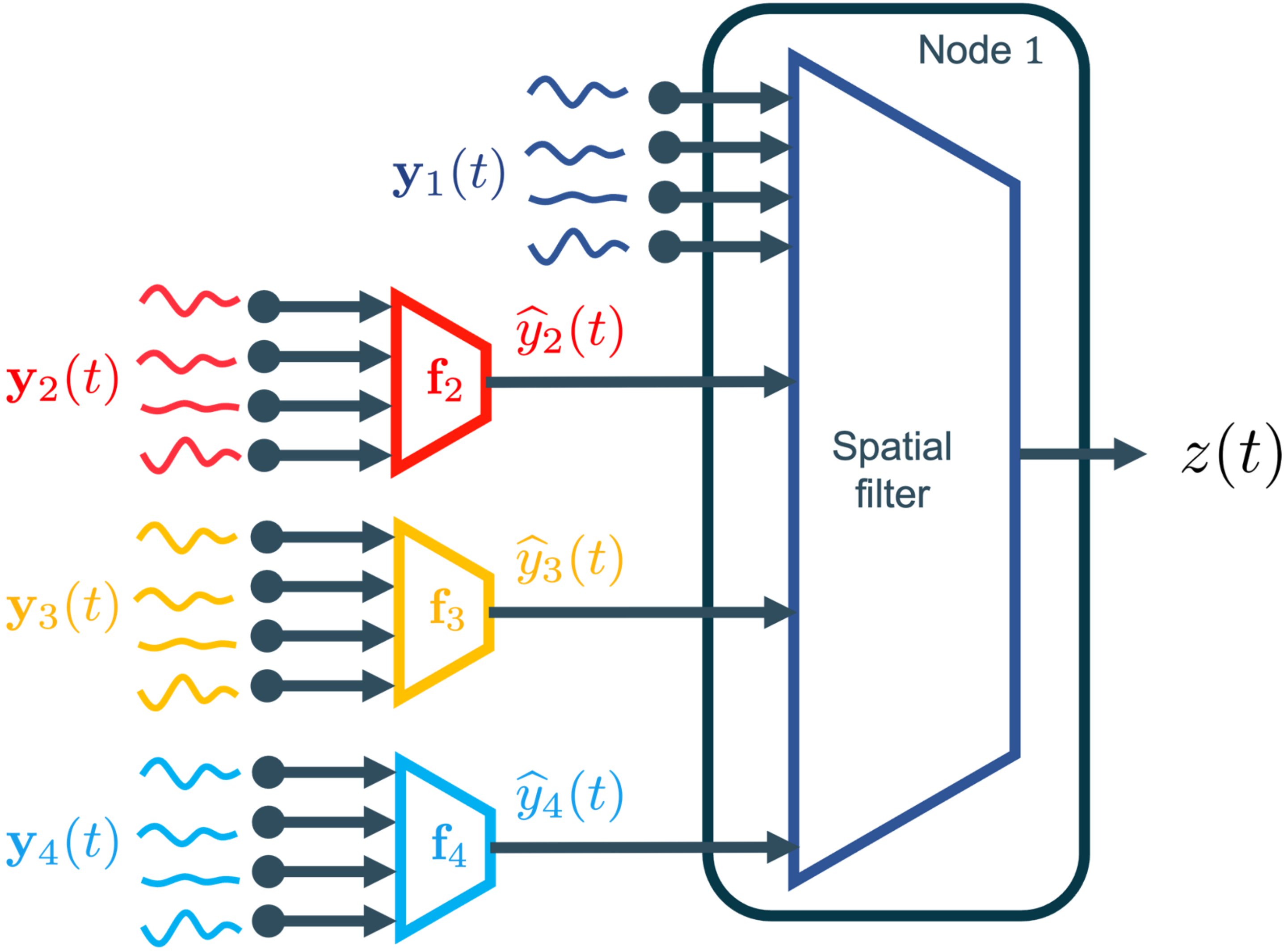


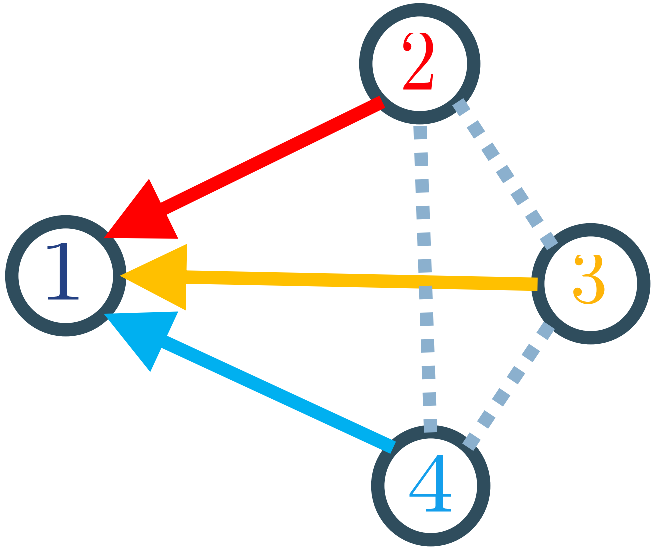


Data flow (2 node example)



Data flow (Fully-connected network)

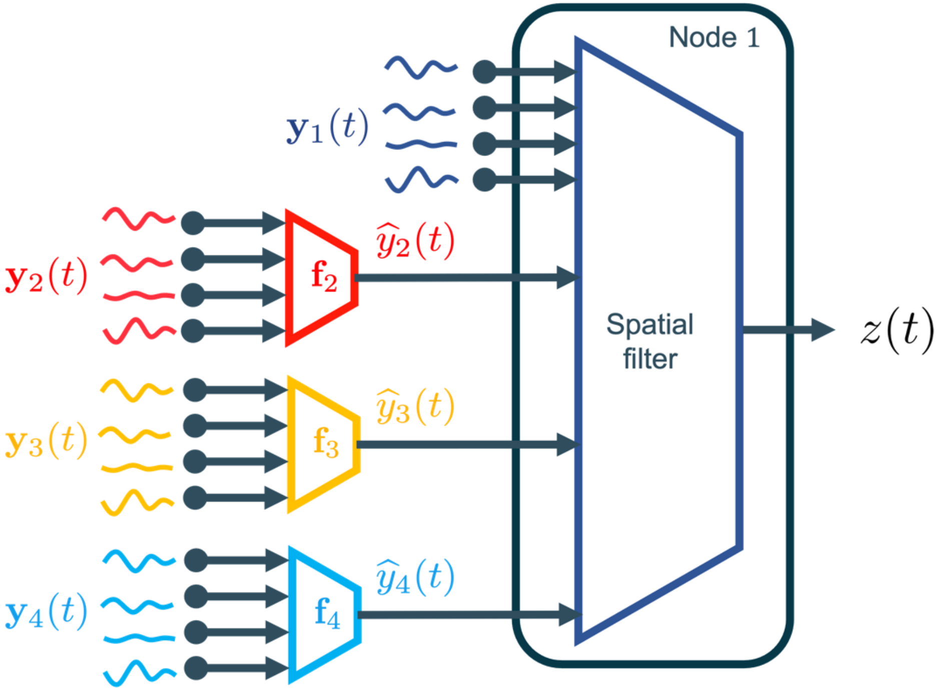




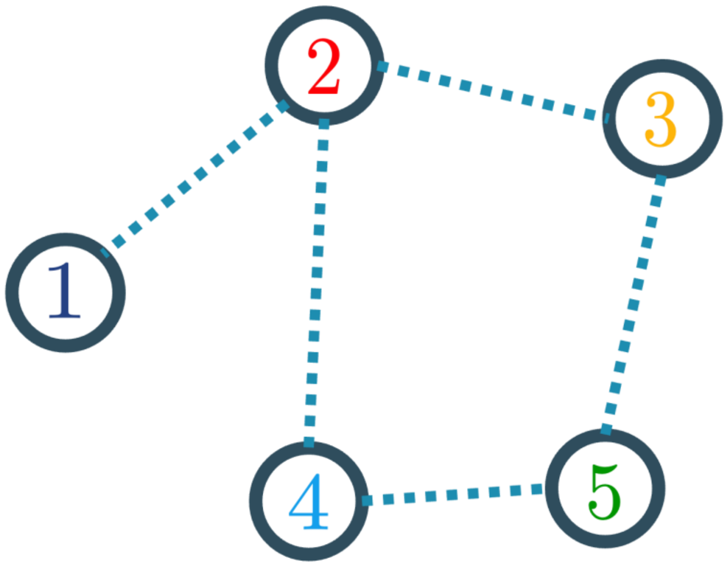
Remember this notation!

$\boldsymbol{y}_k$ = **original** (raw) data

$\widehat{\boldsymbol{y}}_k$ = **fused** data



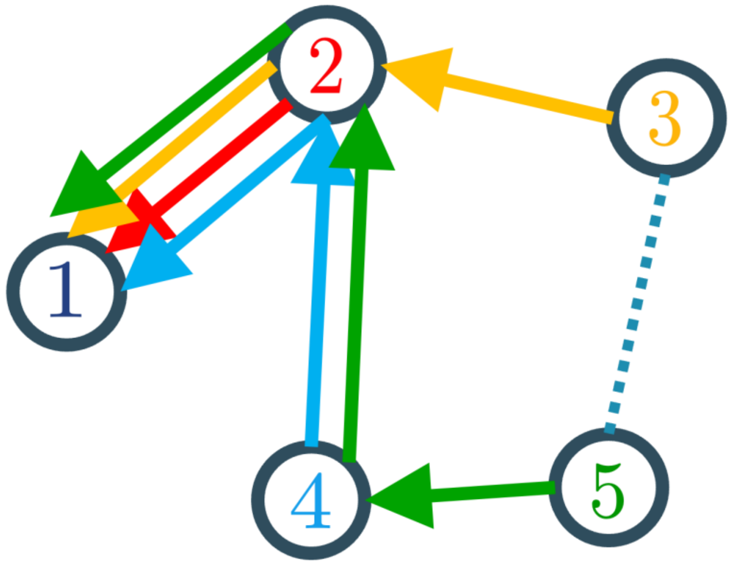
Data flow (General)



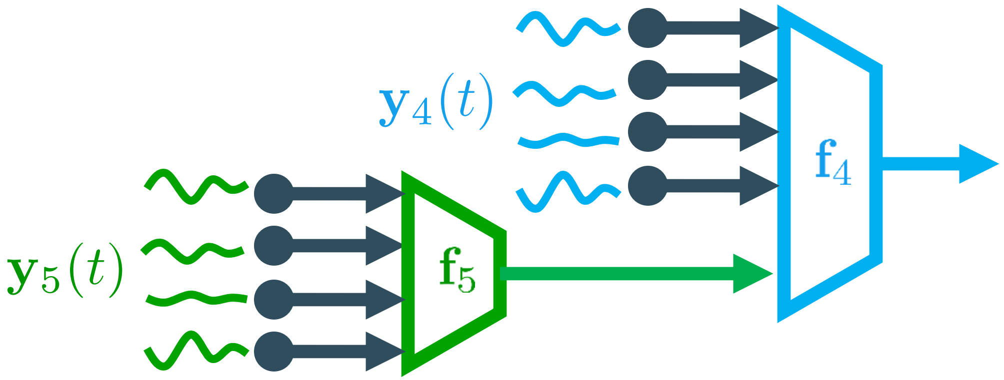
Data flow (General)

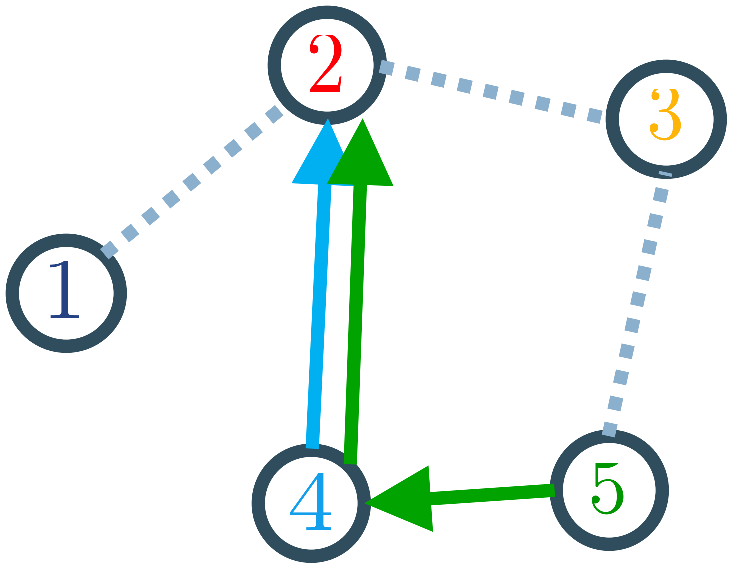
Avoid relaying data !

(not scalable)



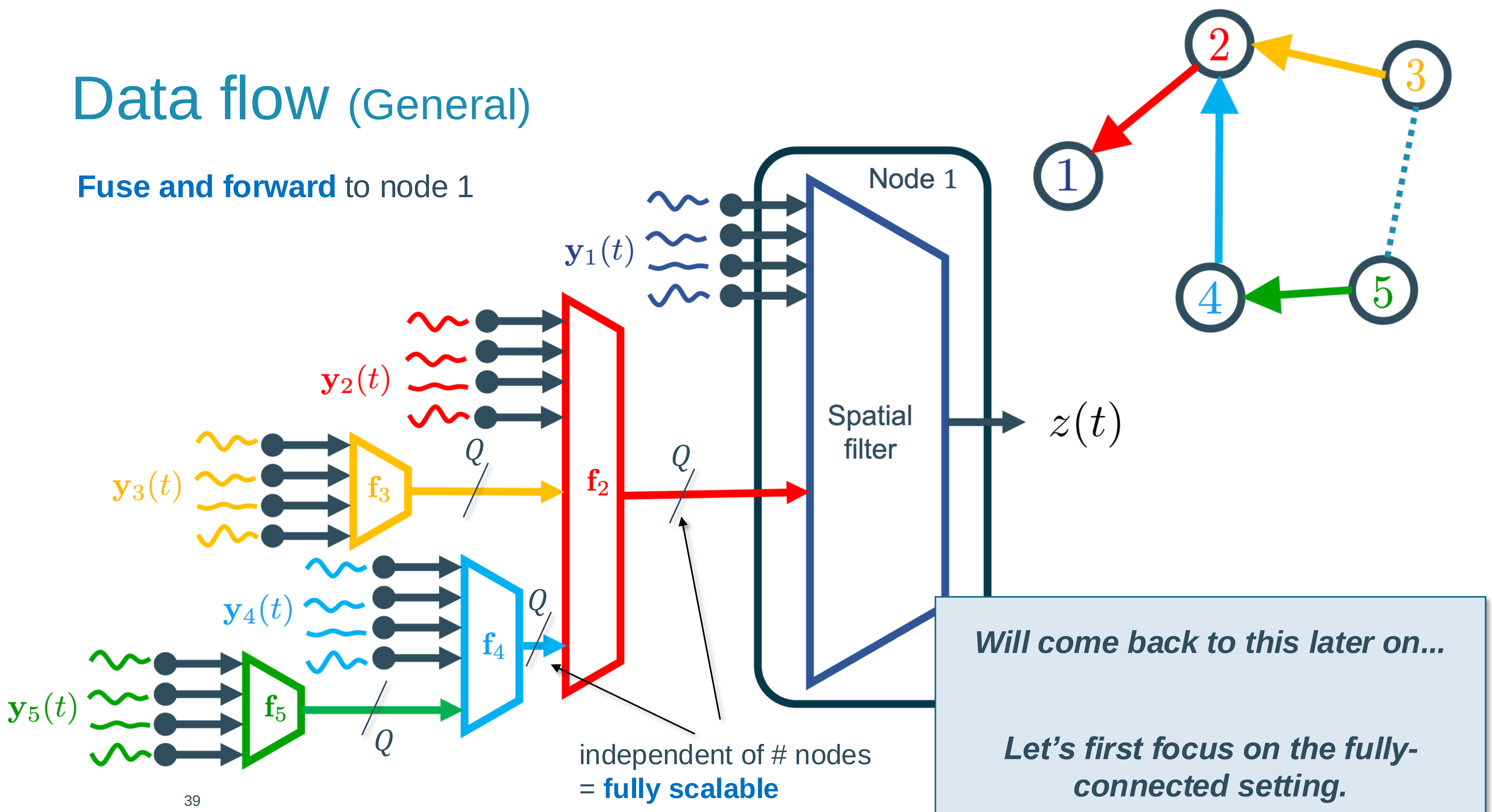
Data flow (General)





Data flow (General)

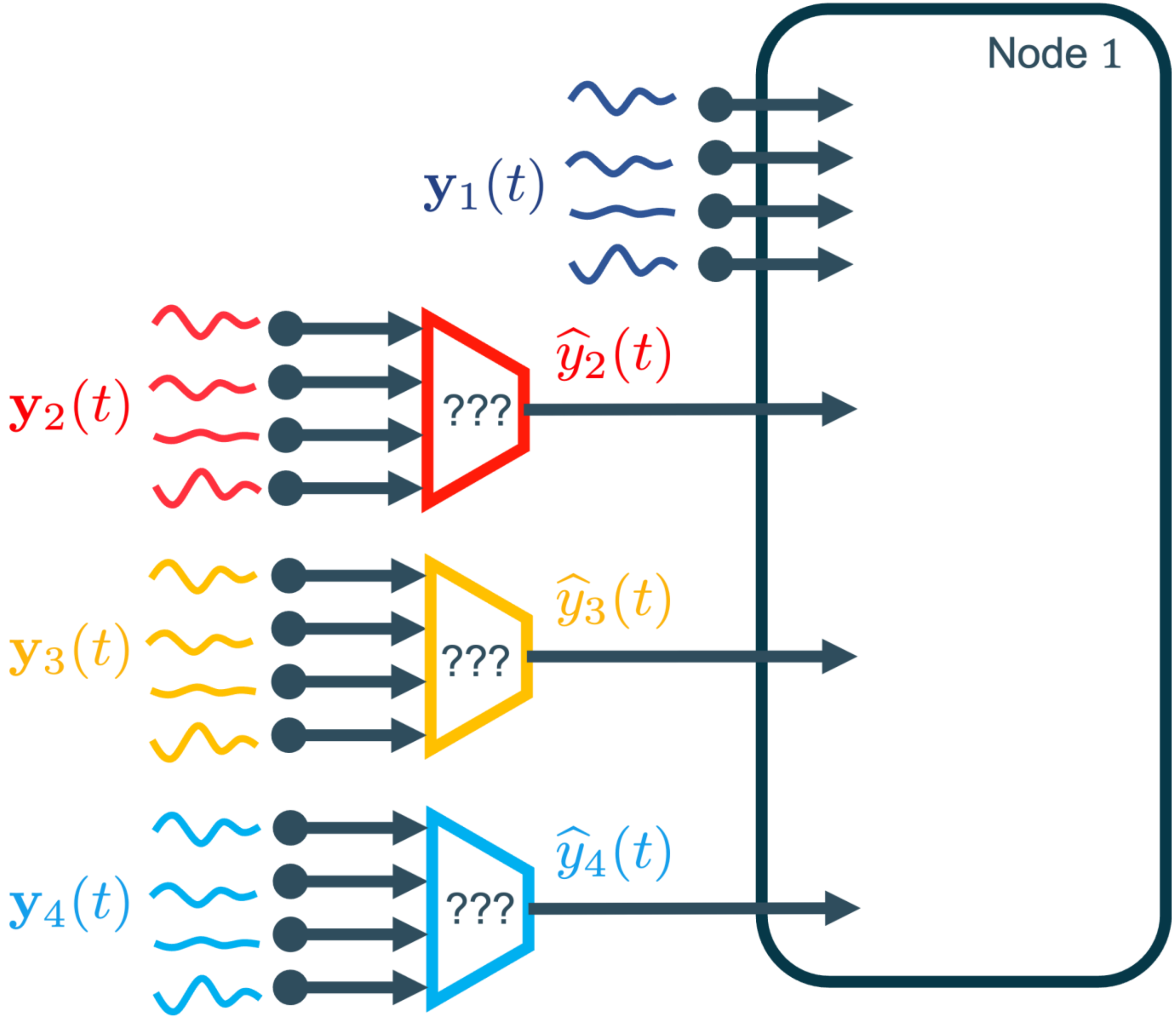
Fuse and forward to node 1

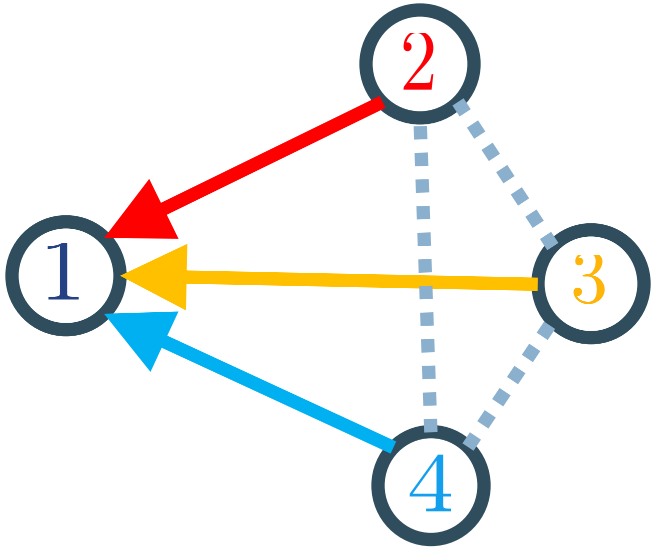


Will come back to this later on…

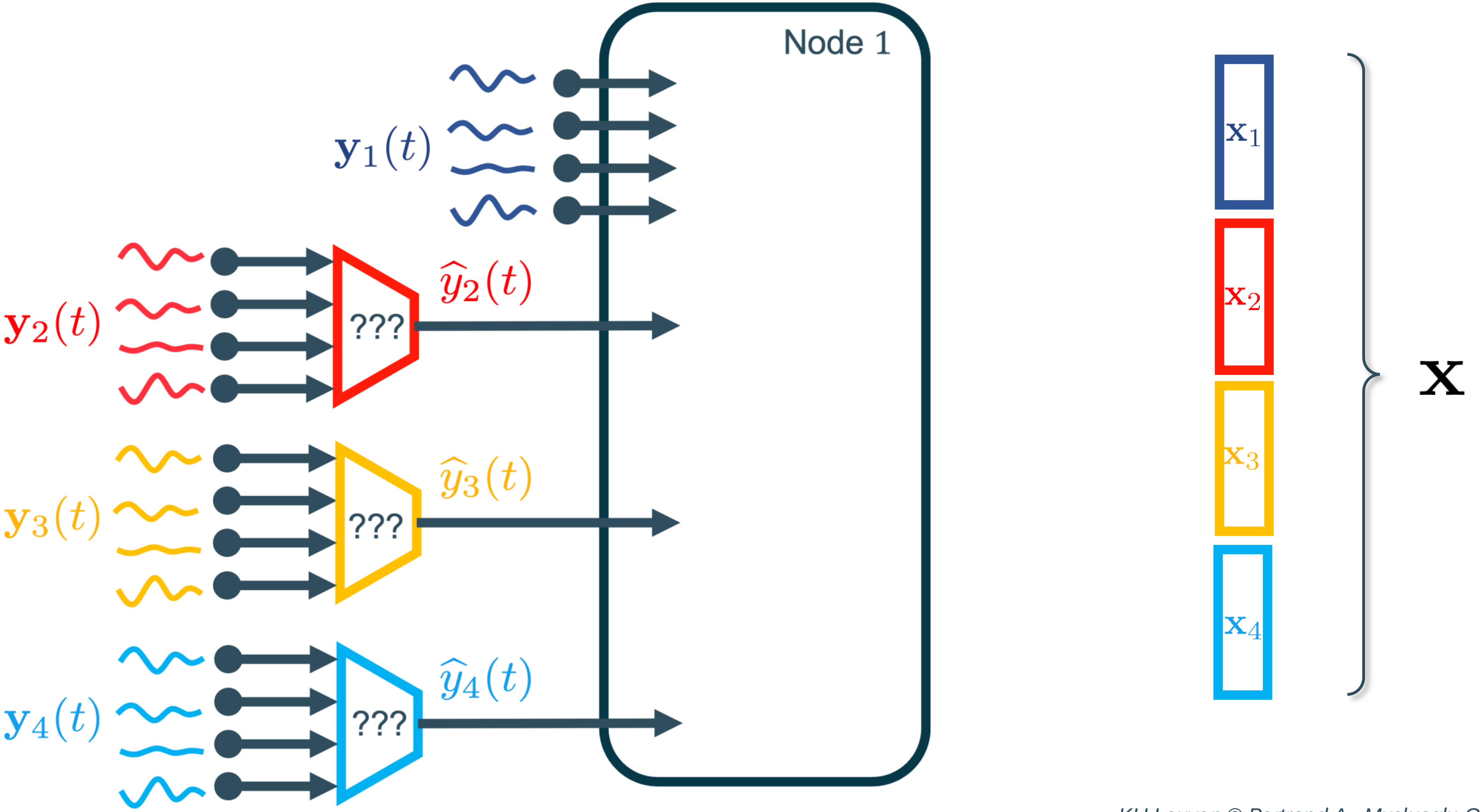
Let's first focus on the fully-connected setting.

Data flow (Fully-connected network)

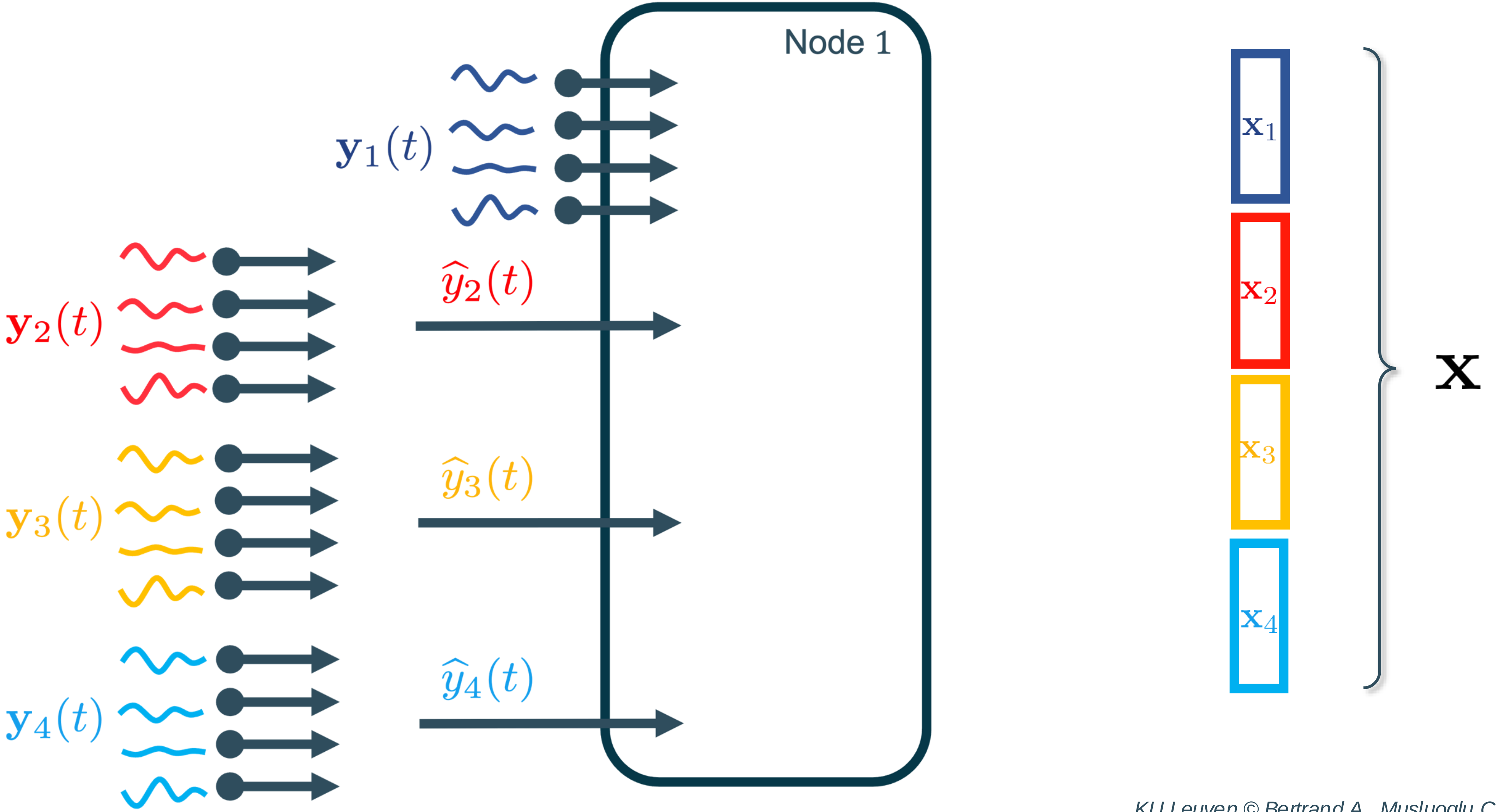




Data flow (Fully-connected network)



Data flow (Fully-connected network)

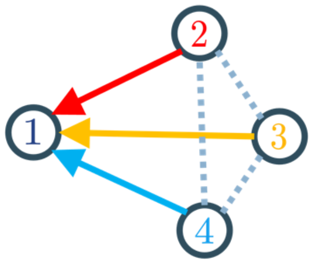


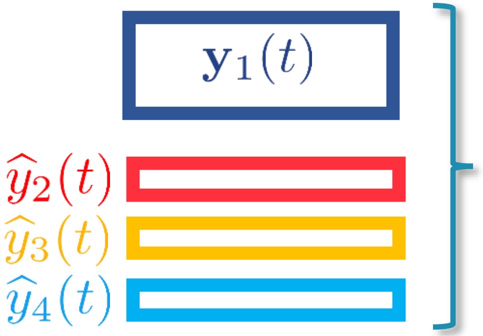
First attempt:
alternating optimization / block-coordinate descent / non-linear Gauss-Seidel

MMSE Example

- Centralized problem: $$\min_{\mathbf{x}} \mathbb{E}\left[\left|d(t) - \mathbf{x}^T\mathbf{y}(t)\right|^2\right] = \min_{\mathbf{x}} \mathbb{E}\left[\left|d(t) - \sum_k \mathbf{x}_k^T\mathbf{y}_k(t)\right|^2\right]$$

- Local MMSE problem at node 1: $$\min_{\mathbf{x}_1} \mathbb{E}\left[\left|d(t) - \mathbf{x}_1^T\mathbf{y}_1(t) - \sum_{k\neq 1} \widehat{y}_k(t)\right|^2\right]$$

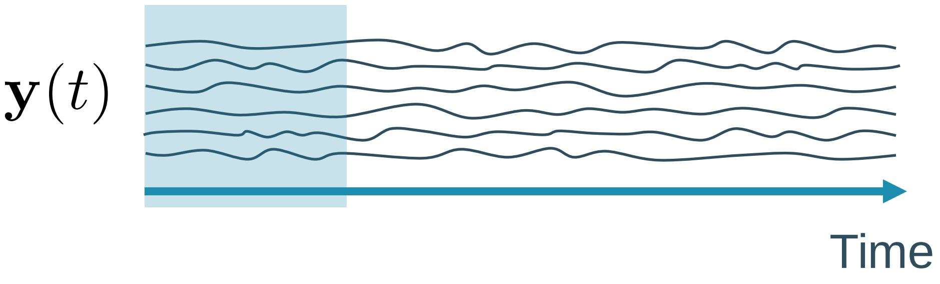


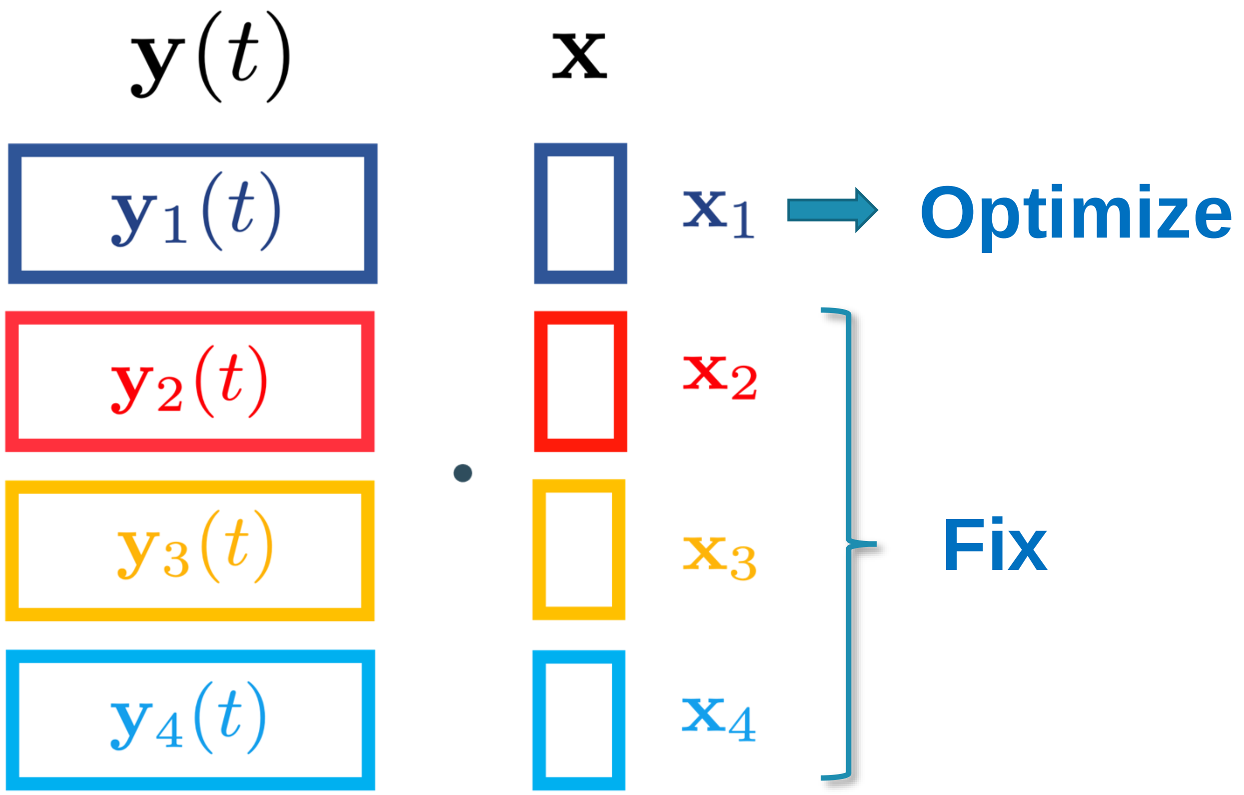


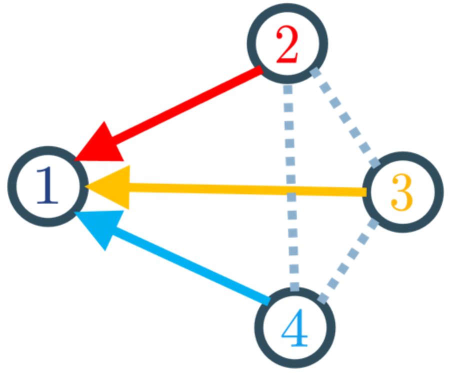
Data available at node 1

First attempt:
alternating optimization / block-coordinate descent / non-linear Gauss-Seidel

- Update in alternating fashion
- Spread iterations over different sample blocks

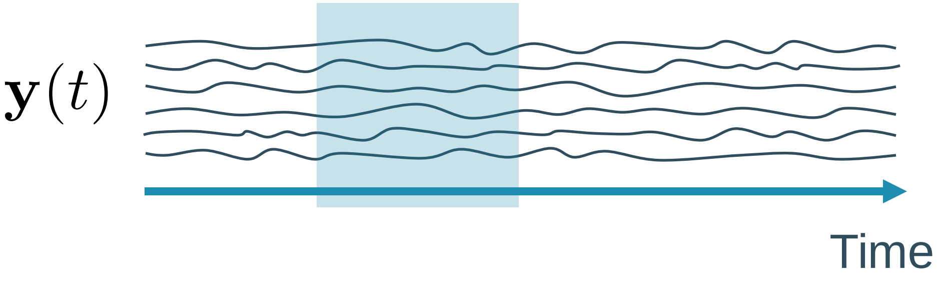


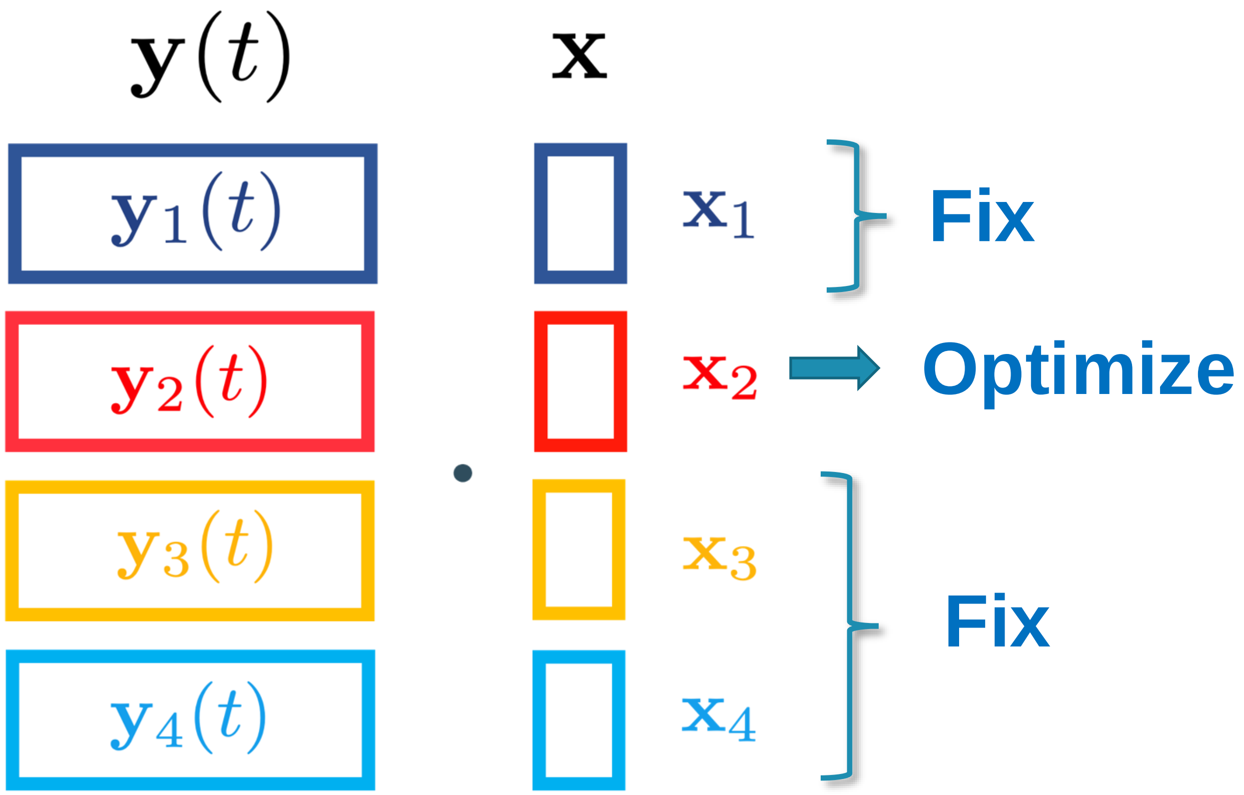


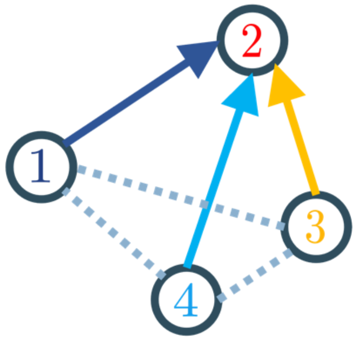


First attempt:
alternating optimization / block-coordinate descent / non-linear Gauss-Seidel

- Update in alternating fashion
- Spread iterations over different sample blocks

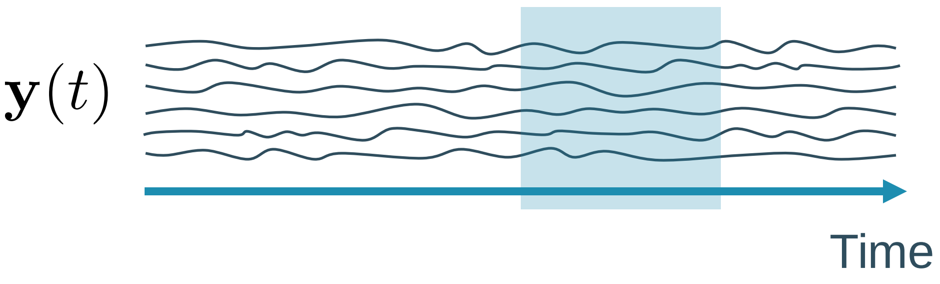


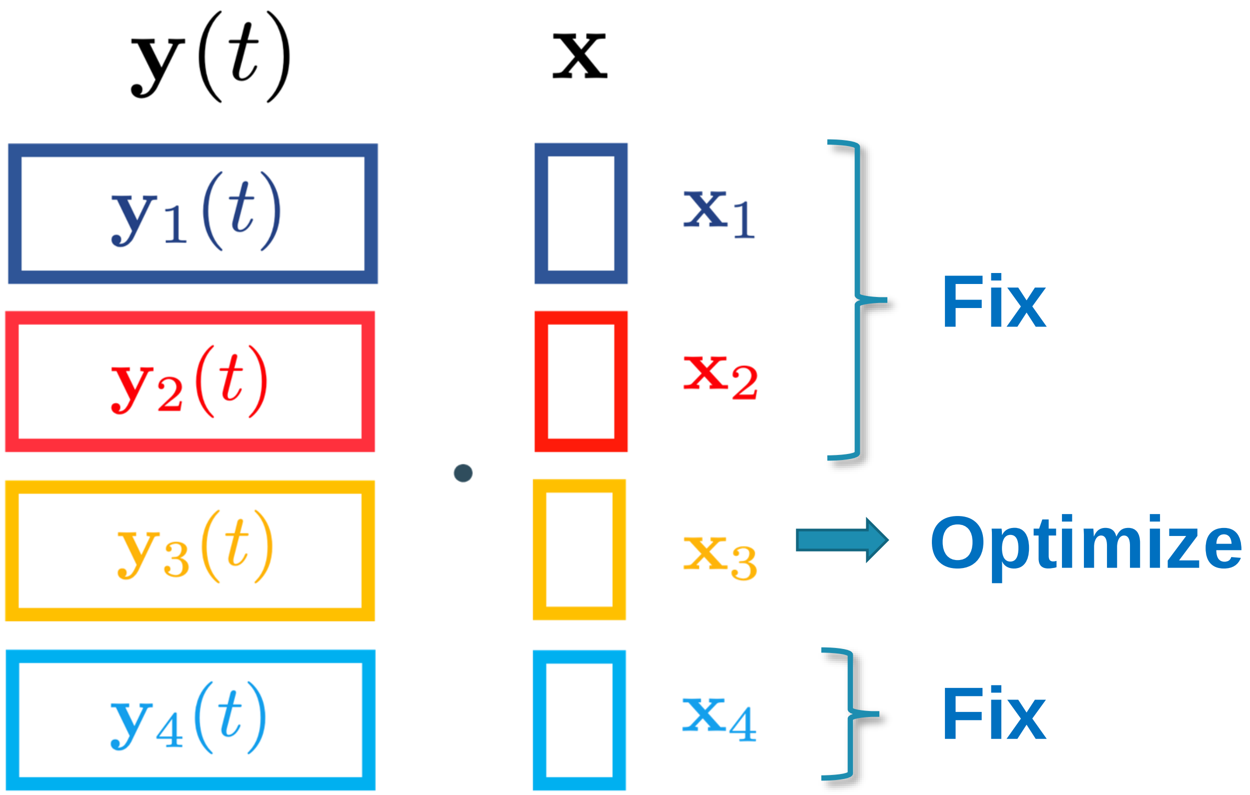


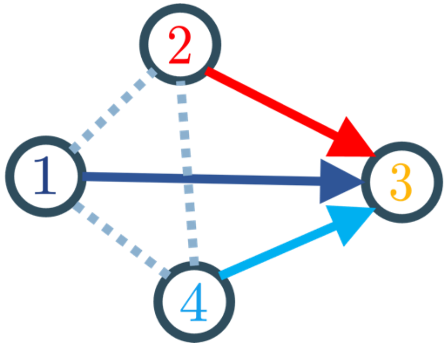


First attempt:
alternating optimization / block-coordinate descent / non-linear Gauss-Seidel

- Update in alternating fashion
- Spread iterations over different sample blocks

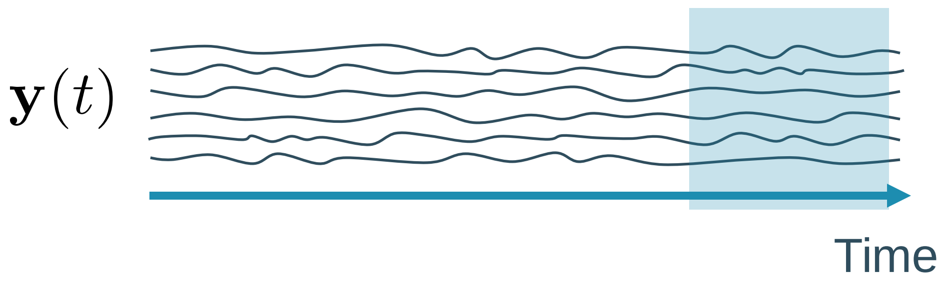


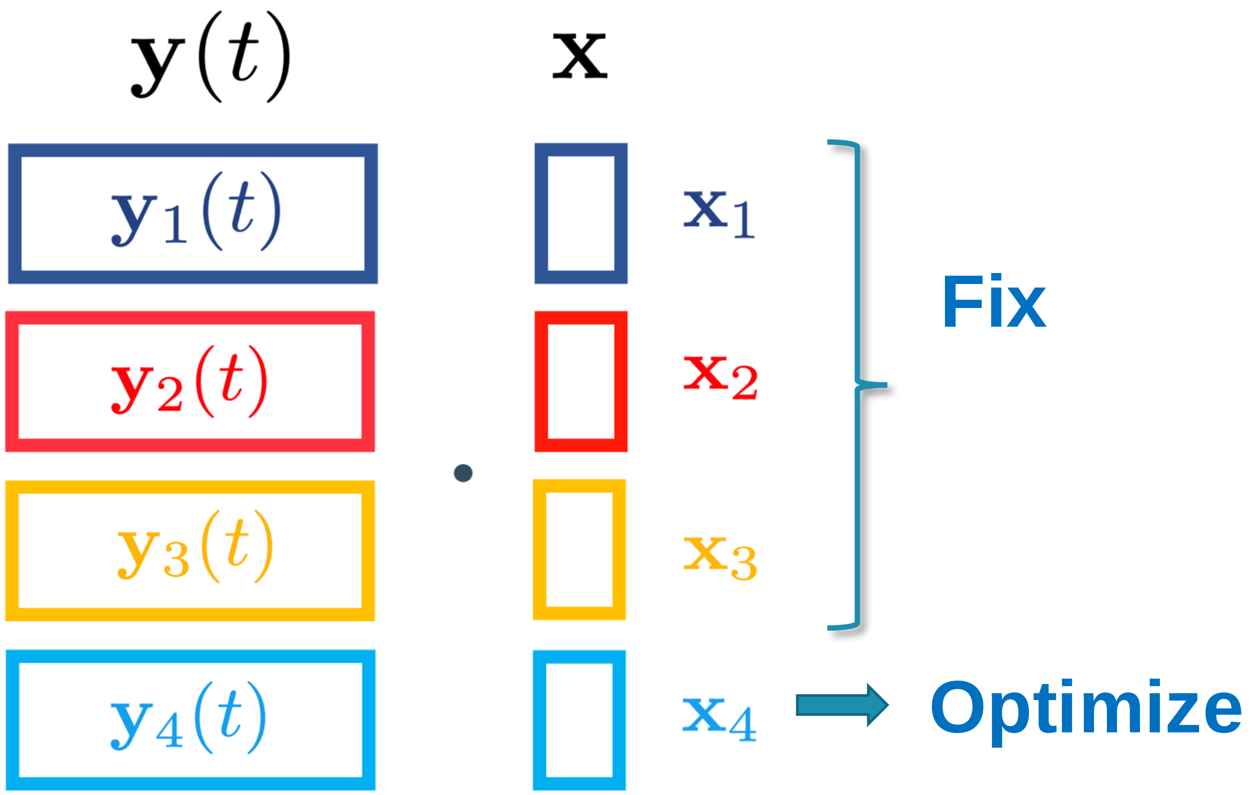


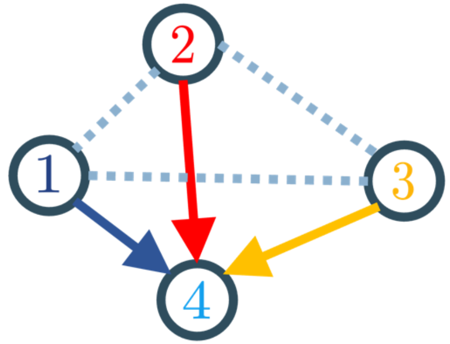


First attempt:
alternating optimization / block-coordinate descent / non-linear Gauss-Seidel

- Update in alternating fashion
- Spread iterations over different sample blocks







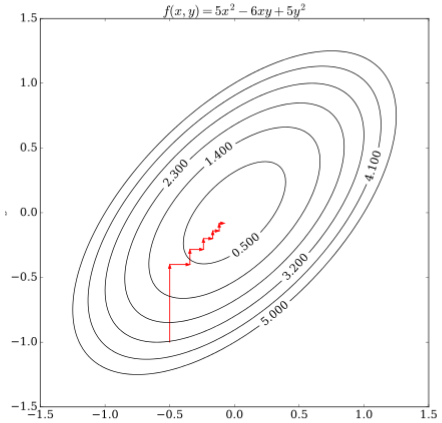
First attempt:
alternating optimization / block-coordinate descent / non-linear Gauss-Seidel

Two problems with this approach:

1) Fixing all $\boldsymbol{x}_k$'s except one

➔ **inefficient** in moving through optimization landscape

➔ effect worsens for large # nodes



2) Local sub-problems generally have a **different structure** than the original problem, and can be hard(er) to solve (see next slide)

First attempt:
alternating optimization / block-coordinate descent / non-linear Gauss-Seidel

PCA Example

$$\max_{\mathbf{x}} \mathbb{E}[|\mathbf{x}^T\mathbf{y}(t)|^2]$$
$$\text{subject to } \mathbf{x}^T\mathbf{x} = 1$$

Solution is an eigenvalue problem: $R_{\mathbf{yy}}\mathbf{x} = \lambda\mathbf{x}$

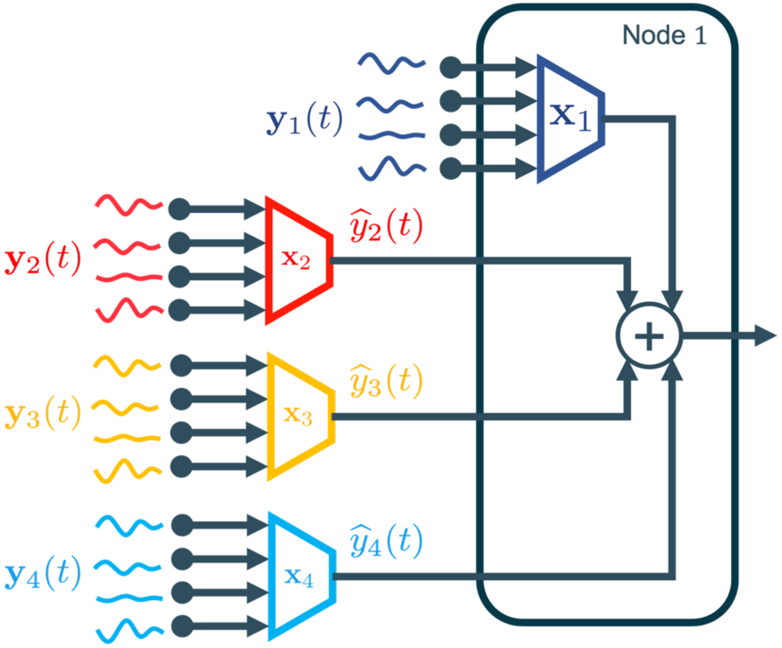
- All $\boldsymbol{x}_k$ are fixed except $\boldsymbol{x}_1$

$$\max_{\mathbf{x}_1} \mathbb{E}[|\mathbf{x}_1^T\mathbf{y}_1(t) + \sum_{k\neq 1}\hat{y}_k(t)|^2]$$
$$\text{subject to } \mathbf{x}_1^T\mathbf{x}_1 = 1 - \sum_{k\neq 1}\mathbf{x}_k^T\mathbf{x}_k$$

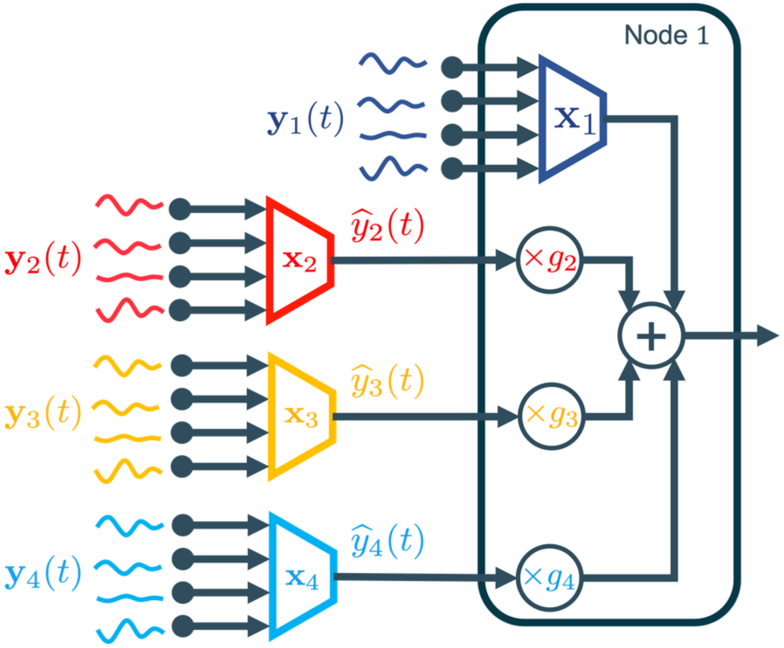
- Solution **not** an eigenvalue problem anymore

$$R_{\mathbf{y}_1\mathbf{y}_1}\mathbf{x}_1 = \lambda\mathbf{x}_1 - \mathbb{E}\left[\mathbf{y}_1(t)\sum_{k\neq 1}\widehat{y}_k(t)\right]$$

Second attempt…



Additional degrees of freedom



Remember this notation!

$\boldsymbol{\theta}_k$ = **original** (raw) data / variable

$\widehat{\boldsymbol{\theta}}_k$ = **fused** data / variable

$\widetilde{\boldsymbol{\theta}}_k$ = **stack** local $\boldsymbol{\theta}$ with $\widehat{\boldsymbol{\theta}}$ from neighboring node(s)

$$\underbrace{\begin{bmatrix} \mathbf{x}_1^T & \square & \square & \square \end{bmatrix}}_{\tilde{\mathbf{x}}_1^T} \cdot \left.\begin{bmatrix} \mathbf{y}_1(t) \\ \widehat{y}_2(t) \\ \widehat{y}_3(t) \\ \widehat{y}_4(t) \end{bmatrix}\right\} \widetilde{\mathbf{y}}_1(t)$$

Revisiting MMSE example

Local problem at node 1

$$\min_{\widetilde{\mathbf{x}}_1} \mathbb{E}[|d(t) - \widetilde{\mathbf{x}}_1^T \widetilde{\mathbf{y}}_1(t)|^2]$$

$$\Rightarrow \tilde{\mathbf{x}}_1^* = R_{\tilde{\mathbf{y}}_1\tilde{\mathbf{y}}_1}^{-1} \mathbb{E}\left[\tilde{\mathbf{y}}_1(t) d(t)\right]$$

$$\Leftrightarrow$$

Original (**centralized**) problem:

$$\min_{\mathbf{x}} \mathbb{E}\left[\left|d(t) - \mathbf{x}^T \mathbf{y}(t)\right|^2\right]$$

$$\Rightarrow \mathbf{x}^* = R_{\mathbf{yy}}^{-1} \mathbb{E}\left[\mathbf{y}(t) d(t)\right]$$

Local problem has same structure as original centralized problem, with smaller dimensions

Formalization:

II.B- The DASF(*) framework

(*) DASF= distributed adaptive signal fusion

(or if you want: distributed adaptive spatial filtering)

The scope of DASF

$$\underset{\mathbf{x}}{\text{minimize}}\ f(\mathbf{x}^T\mathbf{y}(t))$$

Example

LS / MMSE: Signal estimation

$$\underset{\mathbf{x}}{\text{minimize}}\ \mathbb{E}[||d(t)-\mathbf{x}^T\mathbf{y}(t)||]^2$$

The optimization variable x always appears as an inner product with a signal $y(t)$

The scope of DASF

MIMO spatial filters
(Q output channels)

$$\underset{X}{\text{minimize}}\ f(X^T\mathbf{y}(t))$$

$$X = \begin{bmatrix} | & & | \\ \mathbf{x}(1) & \dots & \mathbf{x}(Q) \\ | & & | \end{bmatrix}$$

Example

LS / MMSE: Signal estimation

$$\underset{X}{\text{minimize}}\ \mathbb{E}[||\mathbf{d}(t) - X^T\mathbf{y}(t)||]^2$$

The optimization variable X always appears as an inner product with a signal $y(t)$

The scope of DASF

Constraints

$$\begin{aligned} \underset{X}{\text{minimize}}\ & f(X^T\mathbf{y}(t)) \\ \text{subject to}\ & g_j(X^T\mathbf{y}(t)) = 0, \\ & h_j(X^T\mathbf{y}(t)) \leq 0 \end{aligned}$$

The optimization variable X always appears as an inner product with a signal $y(t)$

The scope of DASF

Multiple signal sets

$$\begin{aligned}\underset{X}{\text{minimize}}\ & f(X^T\mathbf{y}(t), X^T\mathbf{v}(t))\\ \text{subject to}\ & g_j(X^T\mathbf{y}(t), X^T\mathbf{v}(t)) = 0,\\ & h_j(X^T\mathbf{y}(t), X^T\mathbf{v}(t)) \leq 0\end{aligned}$$

Example

Max-SNR/GEVD: Dimensionality reduction

$$\begin{aligned}\underset{X}{\text{maximize}}\ & \mathbb{E}[||X^T\mathbf{y}(t)||]^2\\ \text{subject to}\ & \mathbb{E}[X^T\mathbf{n}(t)\mathbf{n}(t)^T X] = I\end{aligned}$$

The optimization variable X always appears as an inner product with a signal $y(t), v(t), \ldots$

The scope of DASF

Multiple variables

$$\underset{(X,W)}{\text{minimize}}\ f(X^T\mathbf{y}(t), W^T\mathbf{v}(t))$$
$$\text{subject to}\ g_j(X^T\mathbf{y}(t), W^T\mathbf{v}(t)) = 0,$$
$$h_j(X^T\mathbf{y}(t), W^T\mathbf{v}(t)) \leq 0$$

Example

CCA: Correlation between data sets

$$\underset{(X,W)}{\text{maximize}}\ \mathbb{E}[\text{tr}(X^T\mathbf{y}(t)\mathbf{v}^T(t)W)]$$
$$\text{subject to}\ \mathbb{E}[X^T\mathbf{y}(t)\mathbf{y}(t)^T X] = I$$
$$\mathbb{E}[W^T\mathbf{v}(t)\mathbf{v}(t)^T W] = I$$

All optimization variables $X, W, \ldots$ always appear as an inner product with a signal $\boldsymbol{y}(t), \boldsymbol{v}(t), \ldots$

The scope of DASF

Deterministic matrices

$$\begin{aligned}\underset{X}{\text{minimize}}\;& f(X^T\mathbf{y}(t), X^TB, X^TC)\\ \text{subject to}\;& g_j(X^T\mathbf{y}(t), X^TB, X^TC) = 0,\\ & h_j(X^T\mathbf{y}(t), X^TB, X^TC) \leq 0\end{aligned}$$

Example

(Robust) minimum variance beamforming

$$\begin{aligned}\underset{X}{\text{minimize}}\;& \mathbb{E}[||X^T\mathbf{y}(t)||]^2\\ \text{subject to}\;& X^TB = H,\\ & \text{tr}(X^TX) \leq \alpha^2\end{aligned}$$

$$X^TX = (X^T \cdot I)(X^T \cdot I)^T = (X^T \cdot C)(X^T \cdot C)^T$$

All optimization variables X, W, ... always appear as an inner product with either:

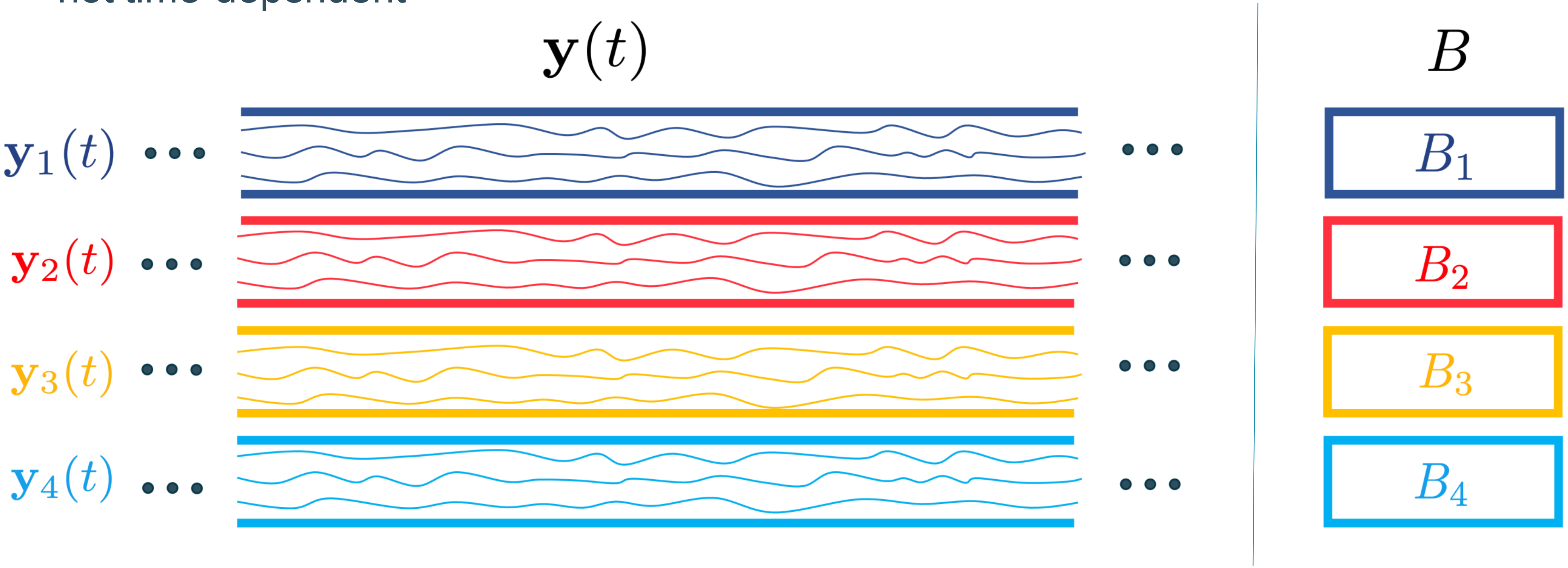
a signal $\boldsymbol{y}(t), \boldsymbol{v}(t), \ldots$

… or a deterministic matrix B, C, ...

C.

The scope of DASF

Deterministic matrices treated similarly as stochastic signal $\boldsymbol{y}$ but are fixed, i.e., not time-dependent



The scope of DASF

$$\underset{X^{(a)},\forall a}{\text{minimize}}\ f\left(X^{(a)T}\mathbf{y}^{(b)}(t), X^{(a)T}B^{(c)}, \dots\right),\ \forall a, b, c$$

$$\text{subject to } h_j\left(X^{(a)T}\mathbf{y}^{(b)}(t), X^{(a)T}B^{(c)}, \dots\right) \leq 0\ \ \forall j \in \mathcal{J}_I,$$

$$h_j\left(X^{(a)T}\mathbf{y}^{(b)}(t), X^{(a)T}B^{(c)}, \dots\right) = 0\ \ \forall j \in \mathcal{J}_E.$$

$a \in \mathcal{A}$: Set of all optimization variables

$b \in \mathcal{B}$: Set of all signals

$c \in \mathcal{C}$: Set of all deterministic matrices that interact as in inner product with $X^{(a)}s$

$\mathcal{J}_I$: Set of all inequality constraints

$\mathcal{J}_E$: Set of all equality constraints

All optimization variables $X^{(a)}$ always appear as an inner product with either:

a signal $\boldsymbol{y}^{(b)}(t)$

… or a deterministic matrix $B^{(c)}$

C.

… let's keep it simple today

$$\begin{aligned} \underset{X}{\text{minimize}} \ & f(X^T\mathbf{y}(t), X^T B) \\ \text{subject to} \ & g_j(X^T\mathbf{y}(t), X^T B) = 0, & \forall j \in \mathcal{J}_I, \\ & h_j(X^T\mathbf{y}(t), X^T B) \leq 0 & \forall j \in \mathcal{J}_E. \end{aligned}$$

X always appears as an inner product with :

signal $y(t)$

… or deterministic matrix B

C.

A trick to remember

$$\begin{aligned} \underset{X}{\text{minimize}} \;& f(X^T\mathbf{y}(t), X^TB) \\ \text{subject to} \;& g_j(X^T\mathbf{y}(t), X^TB) = 0, \\ & h_j(X^T\mathbf{y}(t), X^TB) \leq 0 \end{aligned}$$

If X appears alone or in an inner product with itself, e.g.,

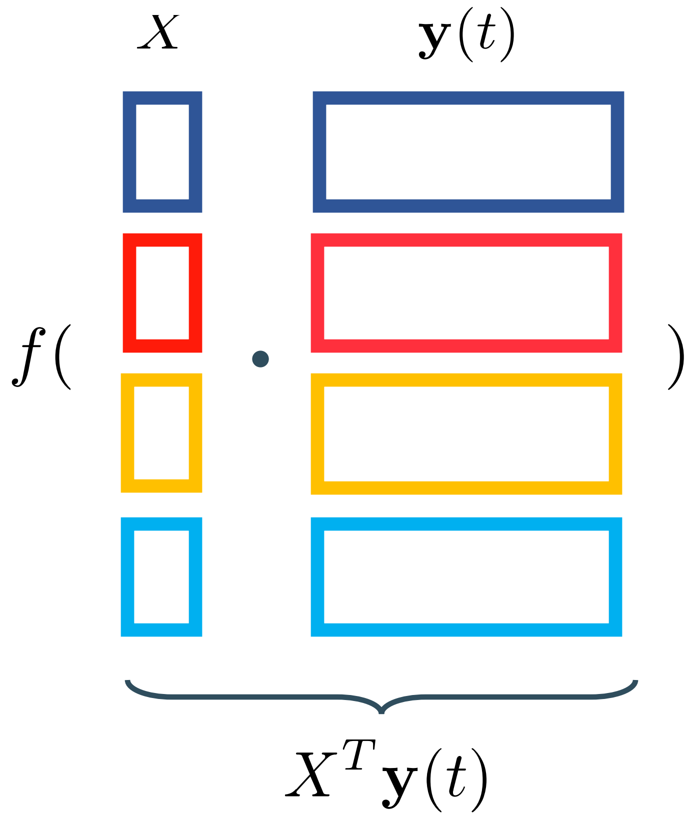
$$\underset{X}{\text{minimize}} \; \|X\|_F$$

(with $B = I$)

$$\underset{X}{\text{minimize}} \; \|X^TB\|_F$$

Functions of sums

Loss and constraints only depend on X through $X^T\boldsymbol{y}(t)$ and possibly on $X^T B$



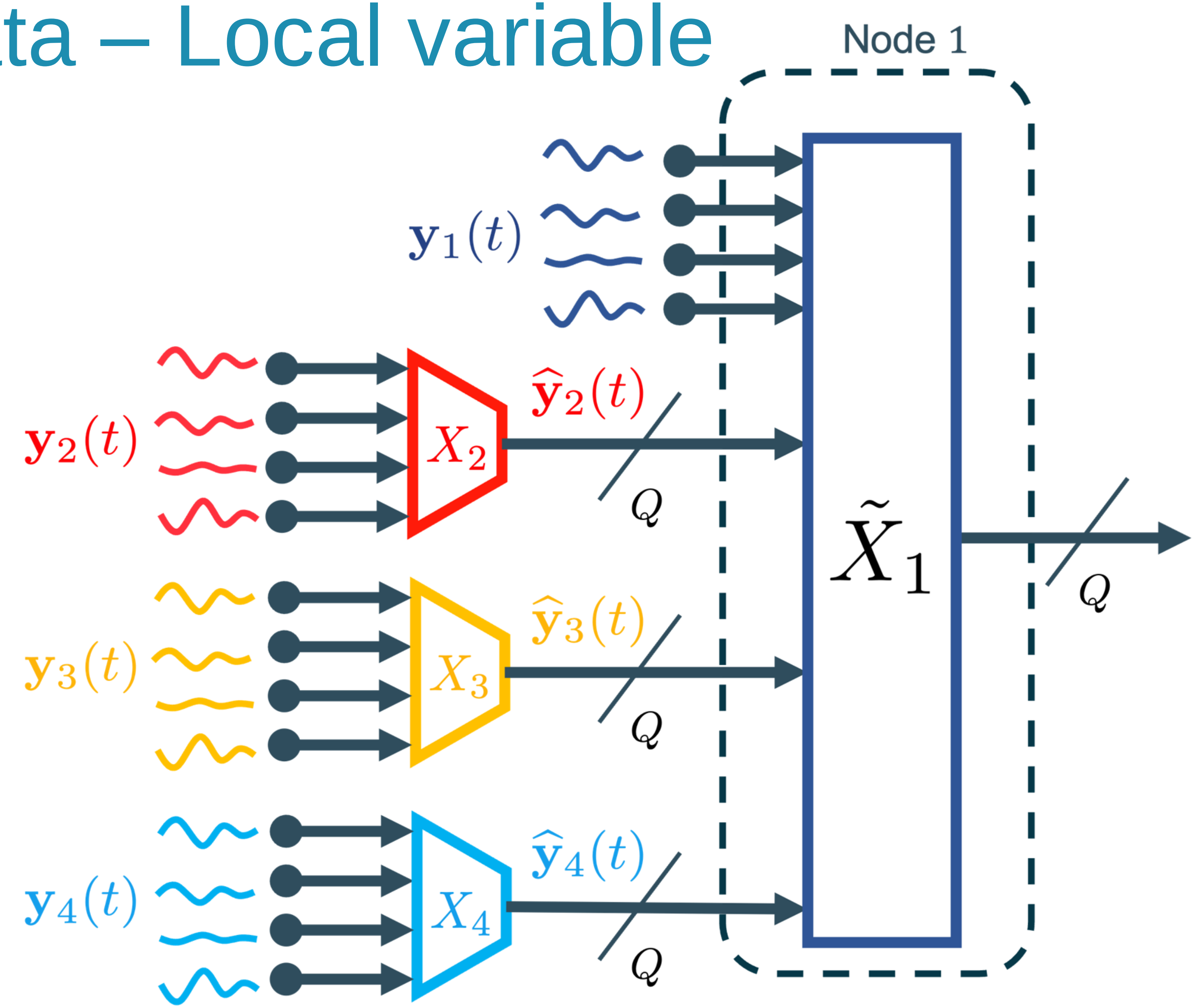
$$= f(\sum_k X_k^T \mathbf{y}_k(t))$$

Functions of sums

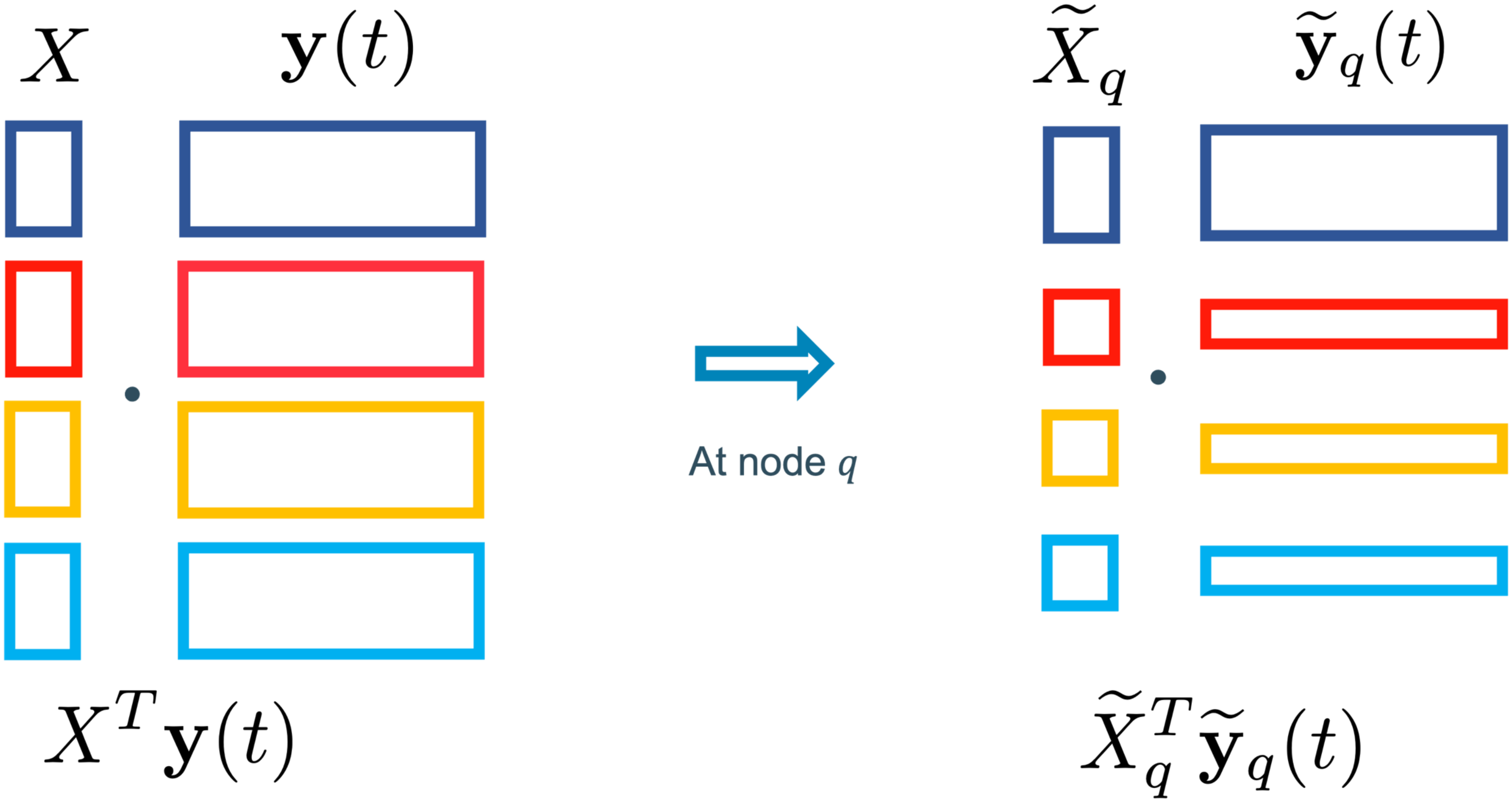
Loss and constraints only depend on X through $X^T \boldsymbol{y}(t)$ and possibly on $X^T B$

$$f(\underbrace{X \cdot B}_{X^T B}) = f(\sum_k X_k^T B_k)$$

Local data – Local variable



Local data – Local variable

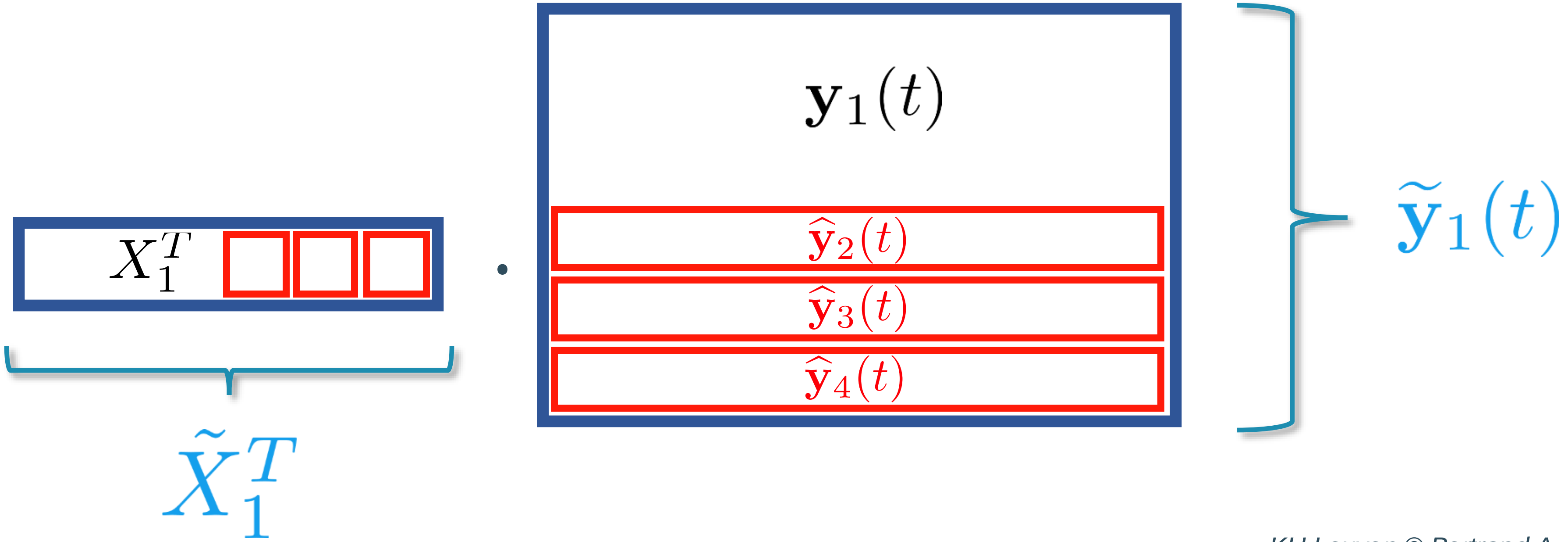


Remember this notation!

$\boldsymbol{\theta}$ = **original** (raw) data / variable

$\widehat{\boldsymbol{\theta}}_{\boldsymbol{k}}$ = **fused** data / variable

$\widetilde{\boldsymbol{\theta}}_{\boldsymbol{k}}$ = **stack** local $\boldsymbol{\theta}$ with $\widehat{\boldsymbol{\theta}}_{\boldsymbol{k}}$ from neighboring node(s)



Local problems ~ Centralized problems

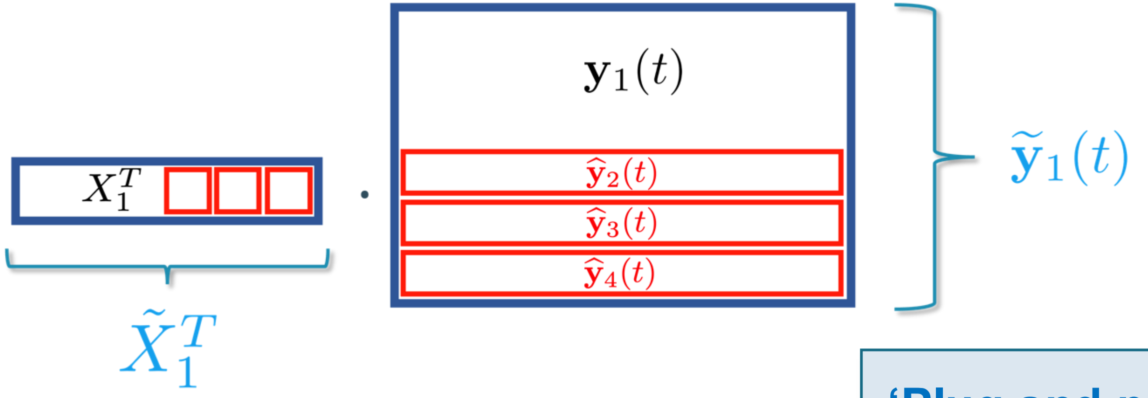
Original (**centralized**) problem:

$$\begin{aligned}\underset{X}{\text{minimize}}\ & f(X^T\mathbf{y}(t), X^T B)\\ \text{subject to}\ & g_j(X^T\mathbf{y}(t), X^T B) = 0,\\ & h_j(X^T\mathbf{y}(t), X^T B) \leq 0\end{aligned}$$

$\Longrightarrow$

Local problem at node *k*

$$\begin{aligned}\underset{\widetilde{X}_k}{\text{minimize}}\ & f(\widetilde{X}_k^T\tilde{\mathbf{y}}_k(t), \widetilde{X}_k^T \widetilde{B}_k)\\ \text{subject to}\ & g_j(\widetilde{X}_k^T\tilde{\mathbf{y}}_k(t), \widetilde{X}_k^T \widetilde{B}_k) = 0,\\ & h_j(\widetilde{X}_k^T\tilde{\mathbf{y}}_k(t), \widetilde{X}_k^T \widetilde{B}_k) \leq 0\end{aligned}$$



‘Plug and play’: Can use the same solver as for original (centralized) problem

Local problems ~ Centralized problems

	Global	Local
MMSE:	$\underset{X}{\text{minimize}}\ \mathbb{E}[\|\|\mathbf{d} - X^T\mathbf{y}\|\|^2]$	$\underset{\widetilde{X}_k}{\text{minimize}}\ \mathbb{E}[\|\|\mathbf{d} - \widetilde{X}_k^T\widetilde{\mathbf{y}}_k\|\|^2]$
LCMV:	$\underset{X}{\text{minimize}}\ \mathbb{E}[\|\|X^T\mathbf{y}\|\|^2]$ $\text{subject to } X^TB = H$	$\underset{\widetilde{X}_k}{\text{minimize}}\ \mathbb{E}[\|\|\widetilde{X}_k^T\widetilde{\mathbf{y}}_k\|\|^2]$ $\text{subject to } \widetilde{X}_k^T\widetilde{B}_k = H$
CCA:	$\underset{X,W}{\text{maximize}}\ \mathbb{E}[\text{tr}(X^T\mathbf{y}\mathbf{v}^TW)]$ $\text{subject to } \mathbb{E}[X^T\mathbf{y}\mathbf{y}^TX] = I_Q$ $\mathbb{E}[W^T\mathbf{v}\mathbf{v}^TW] = I_Q$	$\underset{\widetilde{X}_k,\widetilde{W}_k}{\text{maximize}}\ \mathbb{E}[\text{tr}(\widetilde{X}_k^T\widetilde{\mathbf{y}}_k\widetilde{\mathbf{v}}_k^T\widetilde{W}_k)]$ $\text{subject to } \mathbb{E}[\widetilde{X}_k^T\widetilde{\mathbf{y}}_k\widetilde{\mathbf{y}}_k^T\widetilde{X}_k] = I_Q$ $\mathbb{E}[\widetilde{W}_k^T\widetilde{\mathbf{v}}_k\widetilde{\mathbf{v}}_k^T\widetilde{W}_k] = I_Q$

Local problems ~ Centralized problems

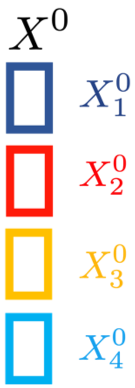
	Global	Local
PCA:	$\underset{\mathbf{x}}{\text{minimize}}\ \mathbb{E}[\lvert\mathbf{x}^T\mathbf{y}(t)\rvert^2]$ subject to $\mathbf{x}^T\mathbf{x}=1$ (with $B=I$) $\Downarrow$ Need to rewrite to fit DASF framework $\underset{\mathbf{x}}{\text{minimize}}\ \mathbb{E}[\lvert\mathbf{x}^T\mathbf{y}(t)\rvert^2]$ subject to $\left(\mathbf{x}^T B\right)\left(B^T\mathbf{x}\right)=1$ $\Rightarrow R_{\mathbf{yy}}\mathbf{x}^*=\lambda(BB^T)\mathbf{x}^*$ generalized eigenvalue problem	$\underset{\tilde{\mathbf{x}}_k}{\text{minimize}}\ \mathbb{E}[\lvert\tilde{\mathbf{x}}_k^T\tilde{\mathbf{y}}_k(t)\rvert^2]$ subject to $\left(\tilde{\mathbf{x}}_k^T\widetilde{B}_k\right)\left(\widetilde{B}_k^T\tilde{\mathbf{x}}_k\right)=1$ $\Rightarrow R_{\tilde{\mathbf{y}}_k\tilde{\mathbf{y}}_k}\tilde{\mathbf{x}}_k^*=\lambda(\widetilde{B}_k\widetilde{B}_k^T)\tilde{\mathbf{x}}_k^*$ generalized eigenvalue problem

III- The fully-connected DASF algorithm

(FC-DASF)

III.A- Algorithm derivation

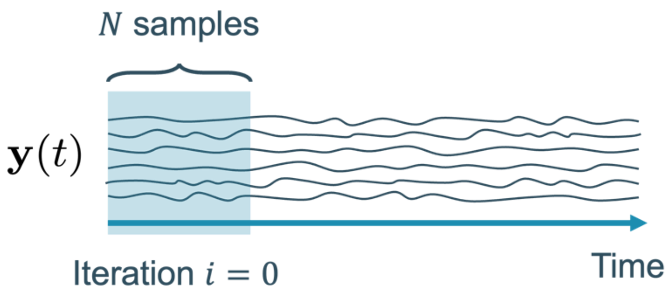
The FC-DASF algorithm

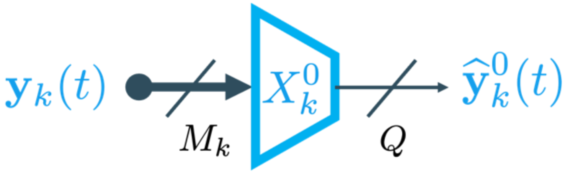


0) Initialization: X^i initialized randomly for $i = 0$

1) Sensor data collection & compression:

- Each node k collects N samples of $\boldsymbol{y}_k$
- Each node k compresses its N samples : $\hat{\boldsymbol{y}}_k^0 = X_k^{0T}\boldsymbol{y}_k$
- … and does the same for B_k (if applicable): $\hat{B}_k^0 = X_k^{0T}B_k$





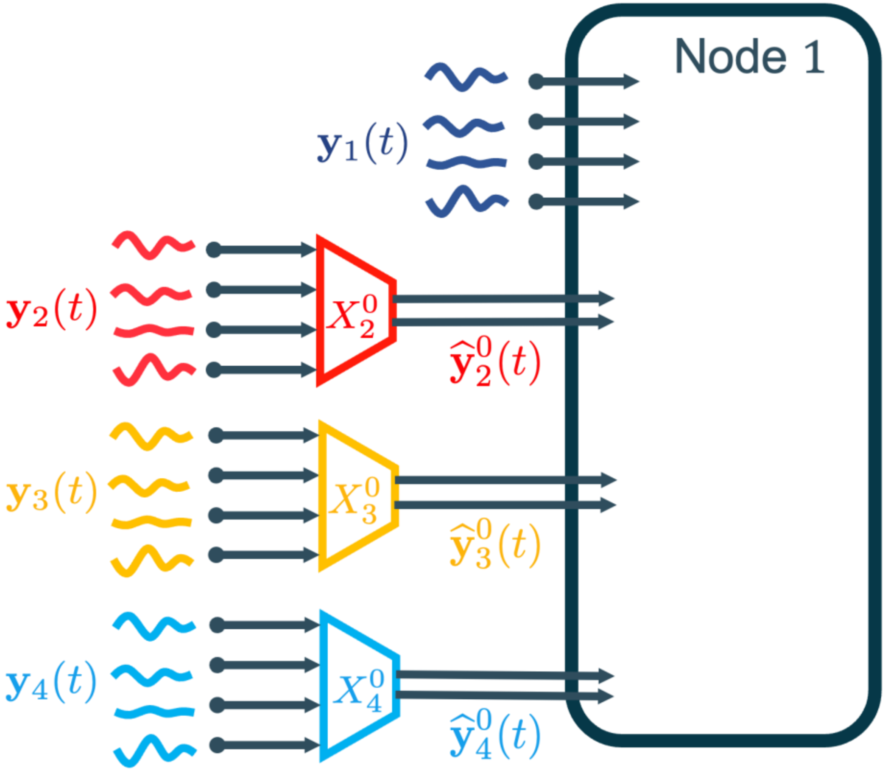
*Note: compression factor is M_k/Q **with Q the #columns of X**, i.e., the number of desired outputs in the spatial filter **(often Q=1)***

The FC-DASF algorithm

2) Transmit compressed data to updating node

- Select new updating node q (in this example $q = 1$)

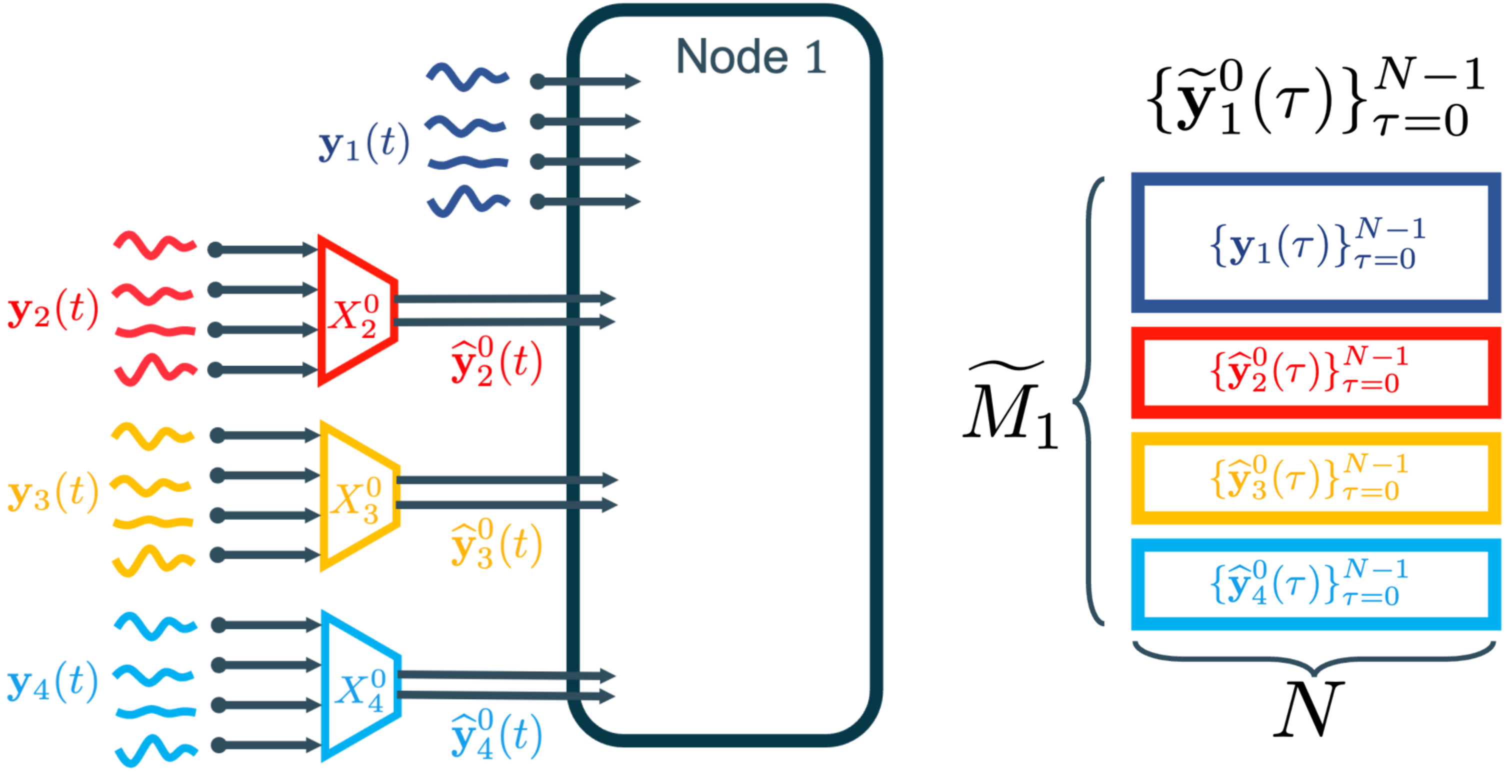
- Nodes $k \neq q$ transmit N samples of $\hat{\boldsymbol{y}}_k^0$ to node q
(+ $\hat{B}_k^0$ if applicable, negligible communication cost)



The FC-DASF algorithm

3) Solve local problem at updating node $q = 1$

- Node $q = 1$ has access to N samples $\{\tilde{y}_q^0(\tau)\}_{\tau=0}^{N-1}$ and to $\tilde{B}_q^0$



$$\widetilde{M}_1 = M_1 + Q(K-1)$$

The FC-DASF algorithm

4) Solve local problem at updating node $q = 1$

- Compute $\widetilde{X}_1^1$ as the solution of

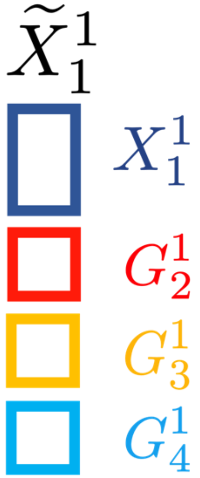
$$
\begin{aligned}
\widetilde{X}_1^1 \triangleq \underset{\widetilde{X}_1}{\operatorname{argmin}} \;& f(\widetilde{X}_1^T \widetilde{\mathbf{y}}_1^0(t), \widetilde{X}_1^T \widetilde{B}_1^0) \\
\text{subject to } \;& g_j(\widetilde{X}_1^T \widetilde{\mathbf{y}}_1^0(t), \widetilde{X}_1^T \widetilde{B}_1^0) = 0, \\
& h_j(\widetilde{X}_1^T \widetilde{\mathbf{y}}_1^0(t), \widetilde{X}_1^T \widetilde{B}_1^0) \leq 0
\end{aligned}
$$

Iteration index i

Updating node index q

‘Plug and play’: Can use the same solver as for original (centralized) problem

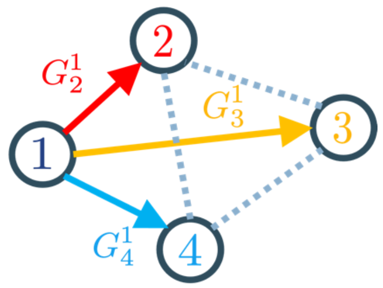
- Partition $\widetilde{X}_1^1$



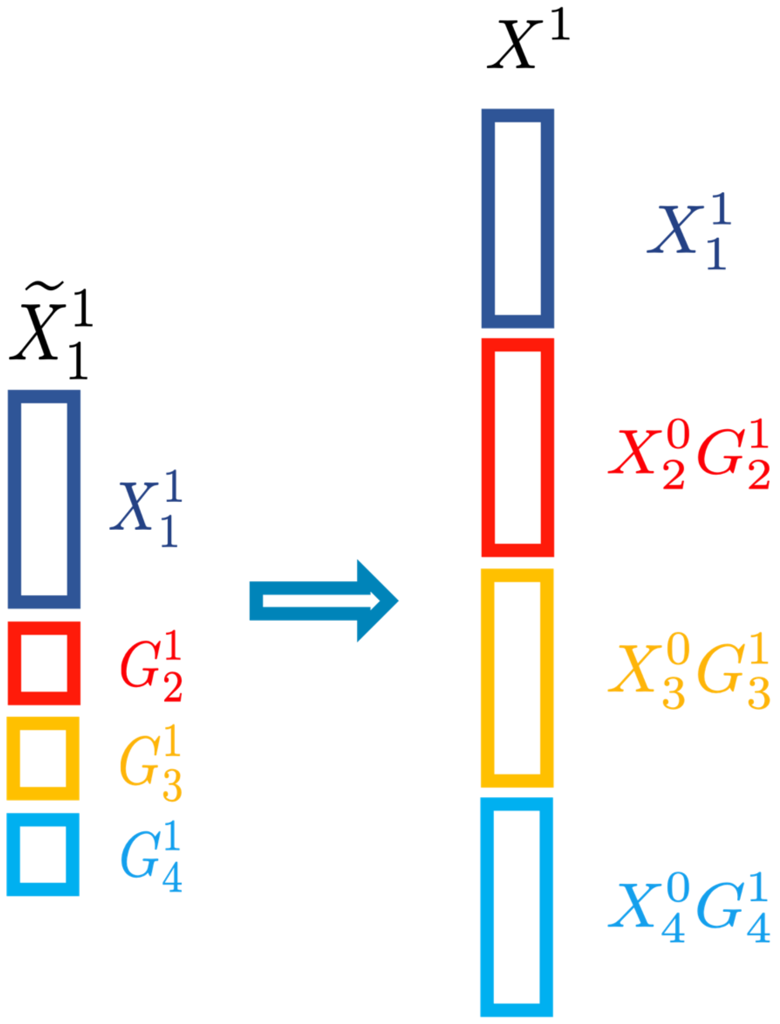
The FC-DASF algorithm

5) Update all nodes

- Node $q = 1$ transmits Q x Q matrices G_k^1



- X^1 is the new estimate of X at iteration $i = 1$

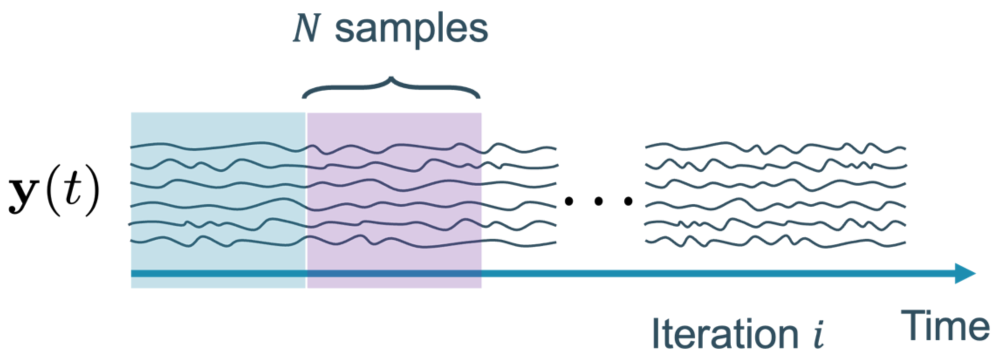


The FC-DASF algorithm

6) If desired: compute output of spatial filter for current block of N samples

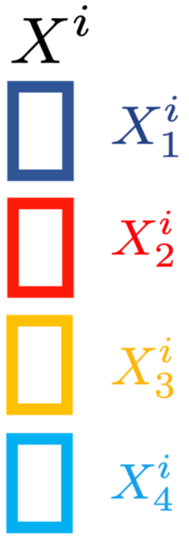
- At updating node $q = 1$: $\left\{\boldsymbol{z}(\tau) = \left(\tilde{X}_q^1\right)^T \tilde{\boldsymbol{y}}_q^0(\tau)\right\}_{\tau=0}^{N-1}$
- If needed: transmit $\{\boldsymbol{z}(\tau)\}_{\tau=0}^{N-1}$ from node q to any node that is a data sink

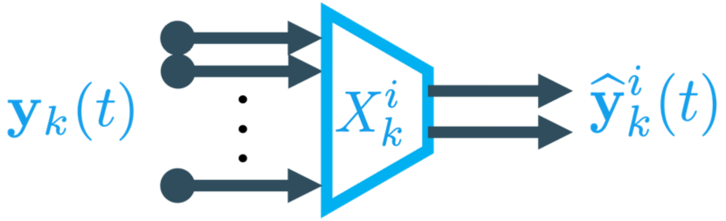
The FC-DASF algorithm



1) Sensor data collection & compression:

- Each node k collects N new samples of $\boldsymbol{y}_k$
- Each node k compresses its N samples : $\widehat{\boldsymbol{y}}_k^i = X_k^{iT}\boldsymbol{y}_k$
- … and does the same for B_k (if applicable): $\widehat{B}_k^i = X_k^{iT}B_k$





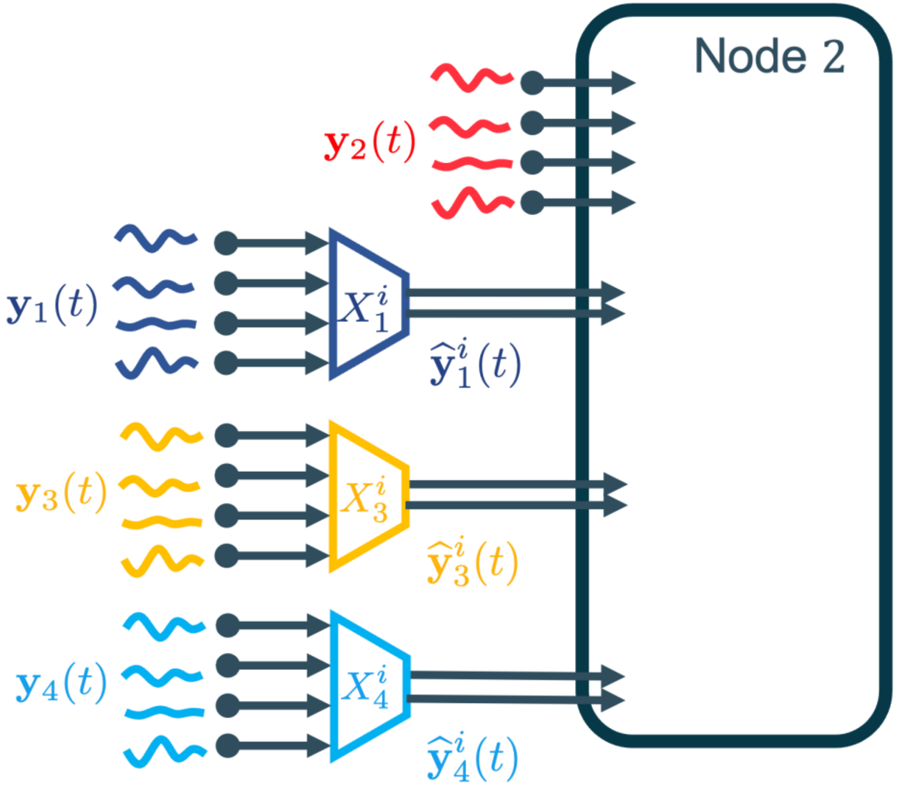
The FC-DASF algorithm

2) Transmit compressed data to updating node

- Select new updating node q (in this example $q = 2$)

- Nodes $k \neq q$ transmit N samples of $\hat{\mathbf{y}}_k^i$ to node q

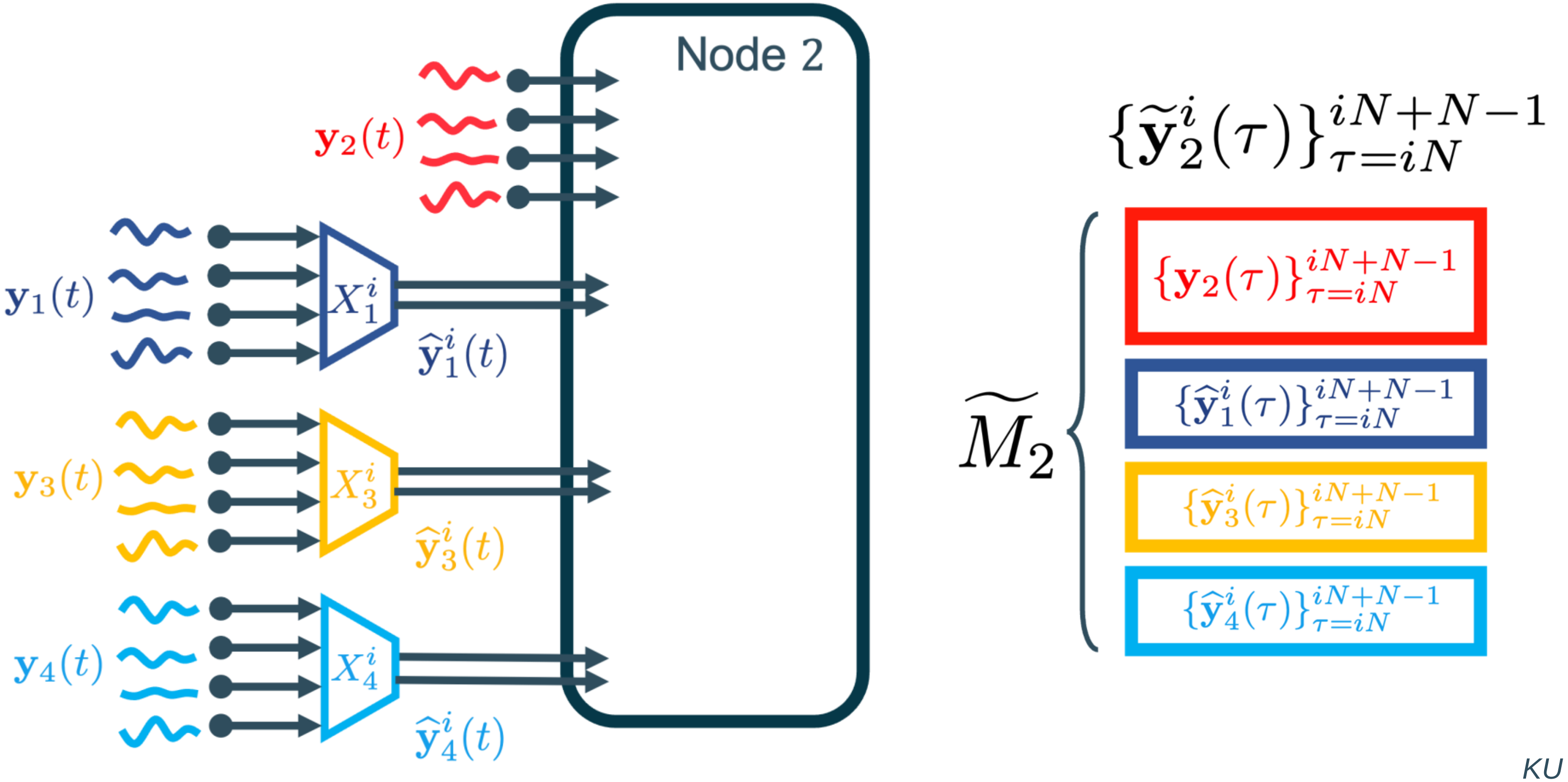
(+ $\hat{B}_k^i$ if applicable)



The FC-DASF algorithm

3) Solve local problem at updating node $q = 2$

- Node $q = 2$ has access to N samples $\{\widetilde{\mathbf{y}}_q^i(\tau)\}_{\tau=iN}^{iN+N-1}$ and to $\tilde{B}_q^i$



$$\widetilde{M}_2 = M_2 + Q(K-1)$$

The FC-DASF algorithm

4) Solve local problem at updating node $q = 2$

- Compute $\widetilde{X}_q^{i+1}$ as the solution of

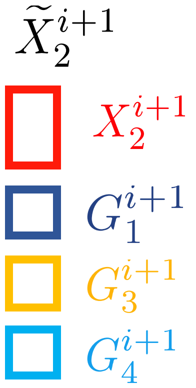
$$
\begin{aligned}
\widetilde{X}_q^{i+1} \triangleq \operatorname*{argmin}_{\widetilde{X}_q} \; & f(\widetilde{X}_q^T \widetilde{\mathbf{y}}_q^i(t), \widetilde{X}_q^T \widetilde{B}_q^i) \\
\text{subject to } & g_j(\widetilde{X}_q^T \widetilde{\mathbf{y}}_q^i(t), \widetilde{X}_q^T \widetilde{B}_q^i) = 0, \\
& h_j(\widetilde{X}_q^T \widetilde{\mathbf{y}}_q^i(t), \widetilde{X}_q^T \widetilde{B}_q^i) \leq 0
\end{aligned}
$$

Iteration index i

Updating node index q

'Plug and play': Can use the same solver as for original (centralized) problem

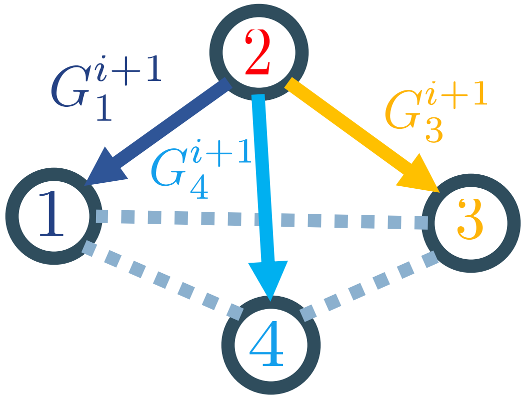
- Partition $\widetilde{X}_2^{i+1}$



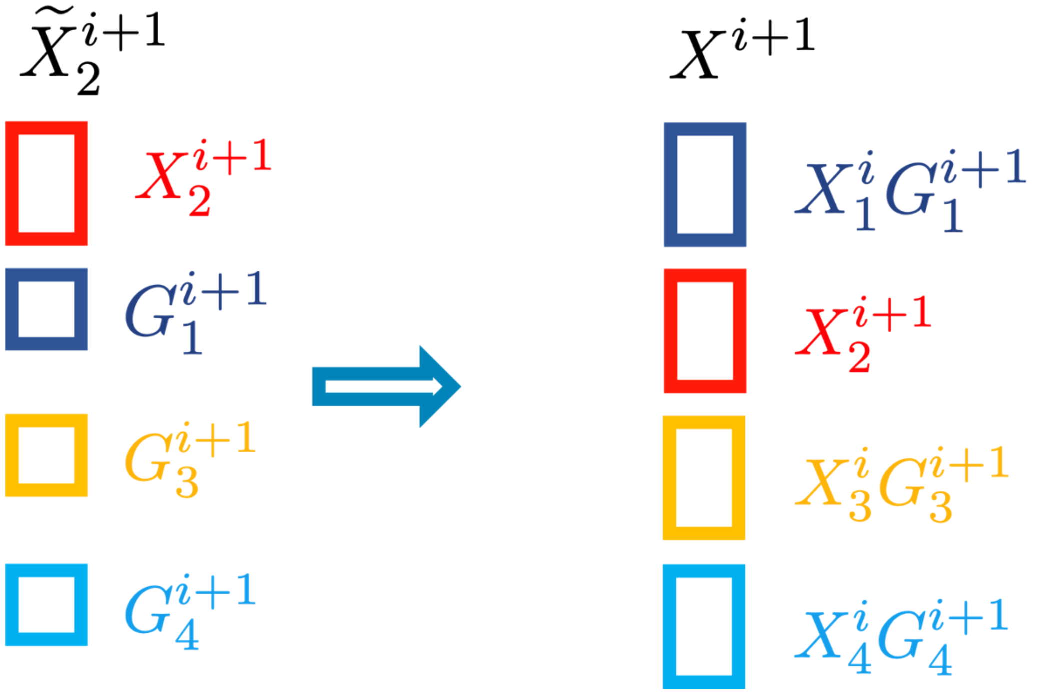
The FC-DASF algorithm

5) Update all nodes

- Node $q = 2$ transmits Q x Q matrices G_k^{i+1}



- X^{i+1} is the new estimate of X at iteration $i+1$



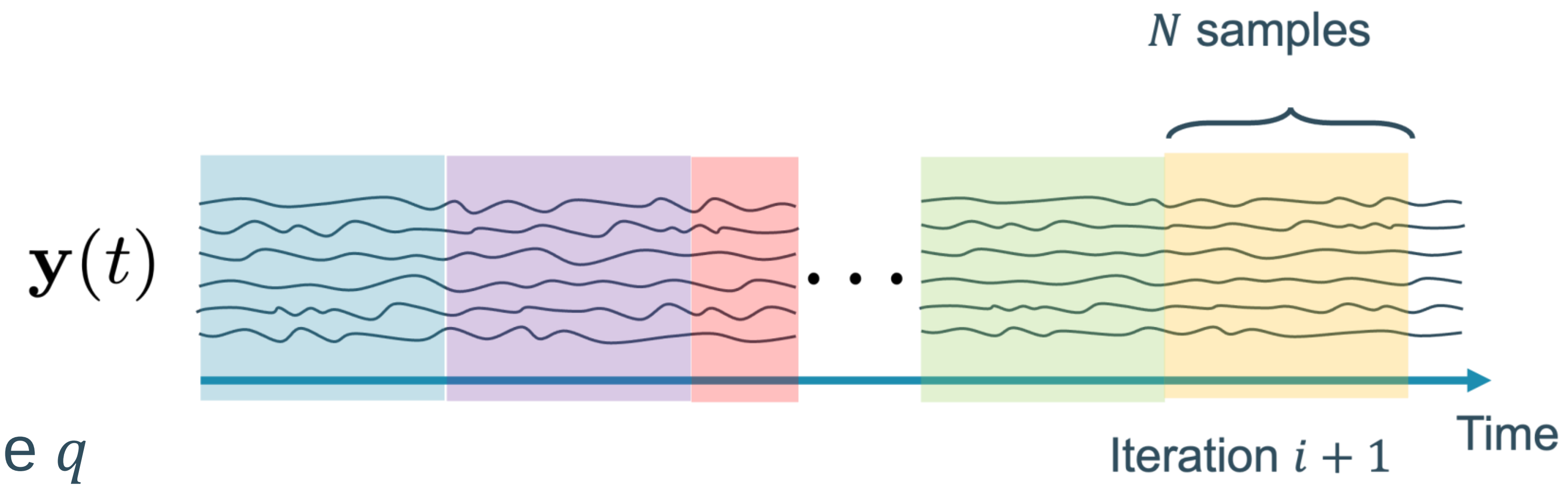
The FC-DASF algorithm

6) If desired: compute output of spatial filter for current block of N samples

- At updating node $q = 2$: $\left\{\boldsymbol{z}(\tau) = \left(\tilde{X}_q^{i+1}\right)^T \tilde{\boldsymbol{y}}_q^i(\tau)\right\}_{\tau=iN}^{iN+N-1}$
- If needed: transmit $\{\boldsymbol{z}(\tau)\}_{\tau=iN}^{iN+N-1}$ from node q to any node that is a data sink

The FC-DASF algorithm

- New samples



- Select new updating node q

- Repeat procedure

Summary of FC-DASF

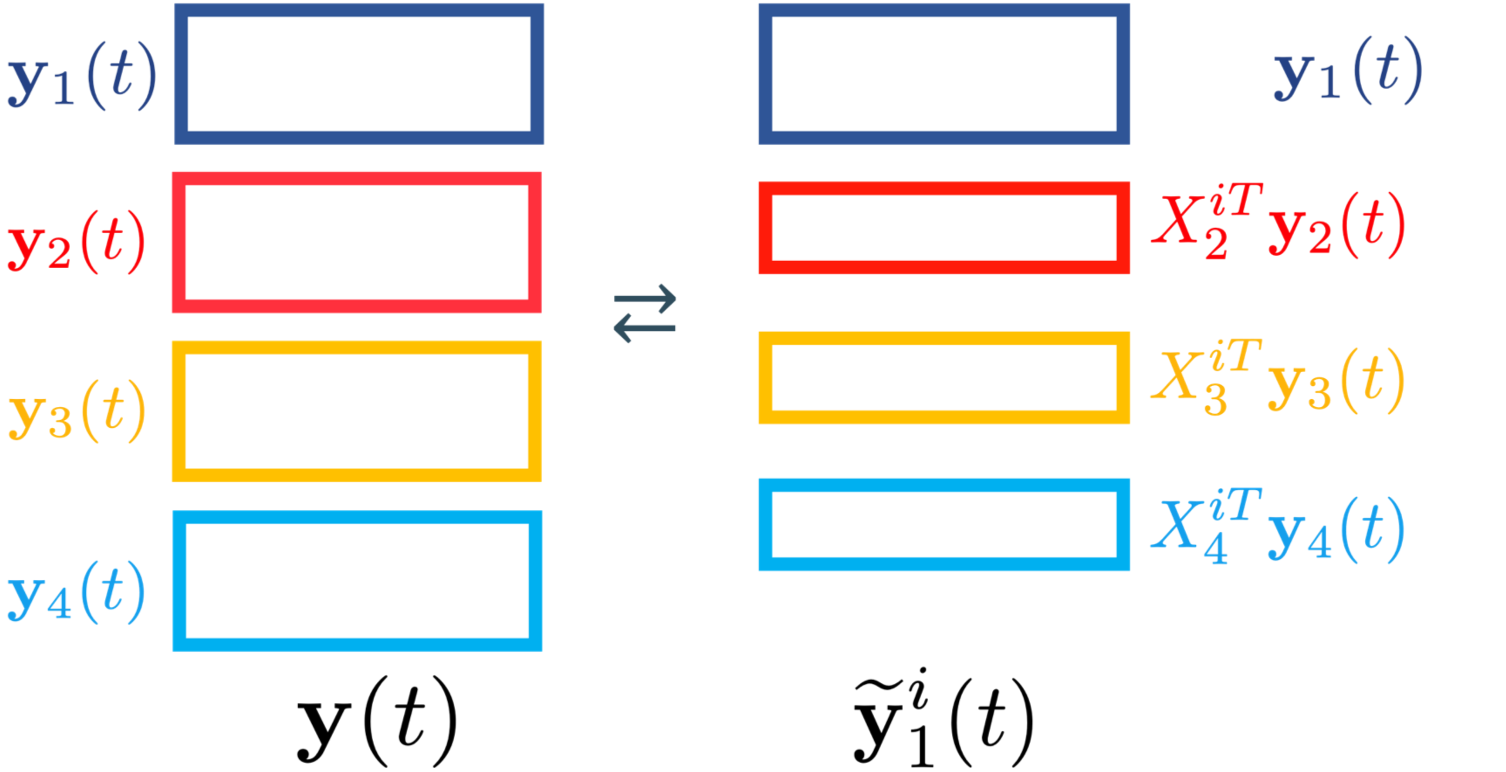
Algorithm 1: Fully-Connected Distributed Adaptive Signal Fusion (FC-DASF) Algorithm

Initialization	Initialize X^0, $i \leftarrow 0$.
	repeat
Select updating node q	Choose the updating node as $q \leftarrow (i \mod K) + 1$.
Collect N new samples, compress and transmit	1) Every node k collects a new batch of N samples of $\mathbf{y}_k$, compresses these to N samples of $\widehat{\mathbf{y}}_k^i$ using $\widehat{\mathbf{y}}_k^i(t) = X_k^{iT}\mathbf{y}_k(t)$ and transmits them to node q.
	$\widehat{B}_k^i$ is computed as $X_k^{iT}B_k$ and transmitted to node q.
	at *Node* q **do**
Solve local problem at node q	2a) Compute $\widetilde{X}_q^{i+1}$ as the solution of the local problem. If the solution is not unique, select the solution which minimizes $\|\widetilde{X}_q^{i+1} - \widetilde{X}_q^i\|_F$ with $\widetilde{X}_q^i = [X_q^{iT}, I_Q, \ldots, I_Q]^T$.
	2b) Partition X_q^{i+1} as $[X_q^T, G_1^T, \ldots, G_{q-1}^T, G_{q+1}^T, \ldots, G_K^T]^T$.
Disseminate G's	2c) Transmit G_k^{i+1} to node k for every $k \neq q$.
	end
Update X	3) Every node updates $X_k^{i+1} = \begin{cases} X_q^{i+1}, & \text{if } q = k, \\ X_k^i G_k^{i+1}, & \text{if } q \neq k. \end{cases}$
	$i \leftarrow i + 1$

III.B- Technical properties

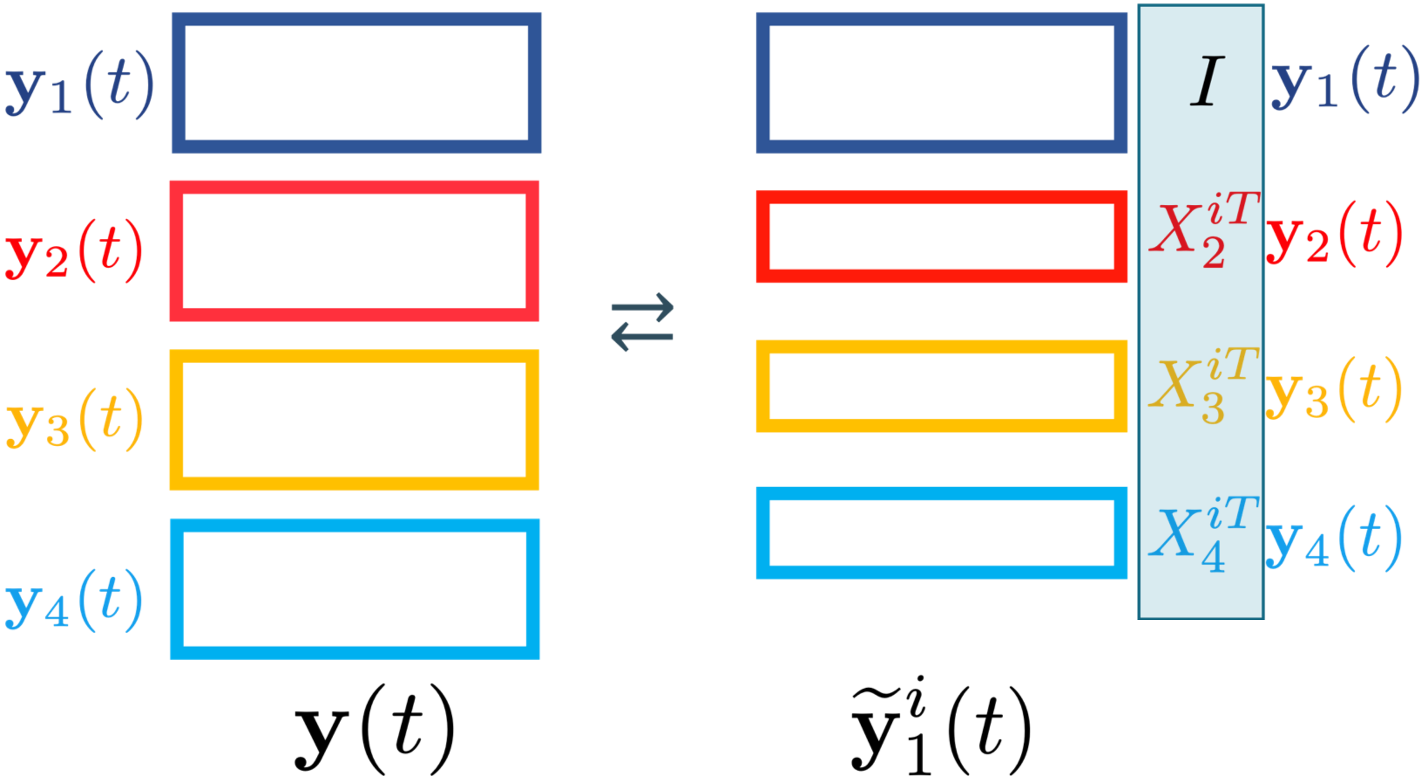
Linear relationship

$$\forall\, i, \forall\, q, \exists\, C_q^i \in R^{M \times \widetilde{M}_q} : \quad C_q^{iT} \mathbf{y}(t) = \tilde{\mathbf{y}}_q^i(t)$$



Linear relationship

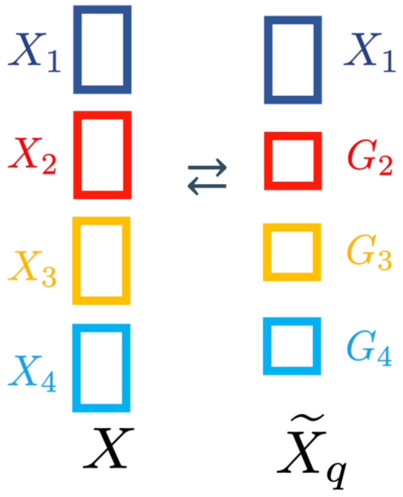
$$\forall\, i, \forall\, q, \exists\, C_q^i \in R^{M \times \widetilde{M}_q} : \quad C_q^{iT} \mathbf{y}(t) = \tilde{\mathbf{y}}_q^i(t)$$



$$C_1^i = blkdiag(I, X_2^i, X_3^i, X_4^i)$$

Linear relationship

Parameterization of variable X via C_q^i : $\quad X = C_q^i \widetilde{X}_q$



Local problem at node q :

$$\begin{aligned} &\underset{\widetilde{X}_q}{\text{minimize}} \; f(\widetilde{X}_q^T \tilde{\mathbf{y}}_q(t), \widetilde{X}_q^T \widetilde{B}_q) \\ &\text{subject to } \widetilde{X}_q \in \widetilde{\mathcal{S}}_q^i \end{aligned} \quad \rightleftarrows \quad \begin{aligned} &\underset{\widetilde{X}_q}{\text{minimize}} \; f(X^T \mathbf{y}(t), X^T B) \\ &\text{subject to } X \in \mathcal{S} \\ &\qquad\qquad\quad X = C_q^i \widetilde{X}_q \end{aligned}$$

$\tilde{\mathcal{S}}_q^i$: Constraint set of the local problem

$\mathcal{S}$: Constraint set of the global problem

Feasible points

- It can be shown that: $\boxed{\widetilde{X}_q \in \widetilde{\mathcal{S}}_q^i \iff X = C_q^i \widetilde{X}_q \in \mathcal{S}}$

- Since $\tilde{X}_q^{i+1} \in \tilde{\mathcal{S}}_q^i$ (by design):

$$X^i \in \mathcal{S}, \forall i > 0$$

i.e., all points generated by DASF are feasible points of the centralized problem

Monotonous decrease in objective

Introducing short notation: $F(X) = f(X^T \boldsymbol{y}(t), X^T B)$

$\left(F(X^i)\right)_i$ is a monotonously decreasing converging sequence

Assumptions

- Objective and constraint functions are **smooth**

  (we will briefly cover the non-smooth setting later)

- The problems are **well-posed**

  i.e., a small change in the parameters of the problems results in a small change in the optimal solution (~ no rank deficient covariance matrices etc.)

- Solutions of the centralized problem are **KKT points**,

  i.e. so-called linearly independent constraint qualifications (**LICQ**) are satisfied

- $\mathcal{S} \cap \{X: F(X) \leq F(X^0)\}$ is **compact**

Fixed points of DASF

- Under some technical conditions (akin to LICQ in optimization literature):

> Any fixed point $\bar{X}$ of DASF is a stationary point of the centralized problem

fixed point = invariant under DASF update, i.e., $X^{i+1} = X^i = \bar{X}$ for any updating node q

- These conditions can be shown to be satisfied ('with high probability') if the number of constraints J is upper bounded as

$$\boxed{J \leq KQ^2}$$

K: Number of nodes in the network
Q: Number of columns of X

Examples

Stiefel manifold constraints: $X^T X = I_Q$

$$X \; is \; M \times Q \qquad \rightarrow J = \frac{Q(Q+1)}{2}$$

$J \leq KQ^2$ always satisfied

Linear constraints: $X^T B = H$

$$B \; is \; M \times L \qquad \rightarrow J = QL$$

$$J \leq KQ^2 \Leftrightarrow L \leq KQ$$

Example: $Q = 2, K = 5 \Longrightarrow L \leq 10$

Convergence of FC-DASF

Under previous assumptions and conditions:
$\left(X^i\right)_i$ converges to a stationary point $\bar{X}$ if the number of 'achievable' solutions is finite

- # 'achievable' solutions here depends on problem <u>and</u> solver:
  - **MMSE** is strongly convex, so only 1 achievable solution by definition
  - **PCA**: $\min_X \operatorname{tr}(X^T R_{\mathbf{yy}} X)$ s. t. $X^T X = I$
    → Infinitely many solutions (if $\bar{X}$ is a solution, then also $\bar{X}U$ with U orth.)
    → Eigenvalue solver extracts the $\bar{X}$ containing the eigenvectors (unique up to sign ambiguities), so # 'achievable' solutions by the solver is finite

- In case of infinite solution set → add a deterministic selection mechanism that leads to a finite set of achievable solutions (e.g. minimum norm, greedy selection, etc.)

Convergence of FC-DASF

Minima are the only **stable** fixed points of DASF

→ Follows from monotonic decrease of the objective

If all minima are global minima:

- **convergence to global solution** in practice (due to instability of all other stationary points)
- holds for many spatial filtering problems (e.g. PCA, CCA, generalized Rayleigh, convex problems, etc.)

Simulations

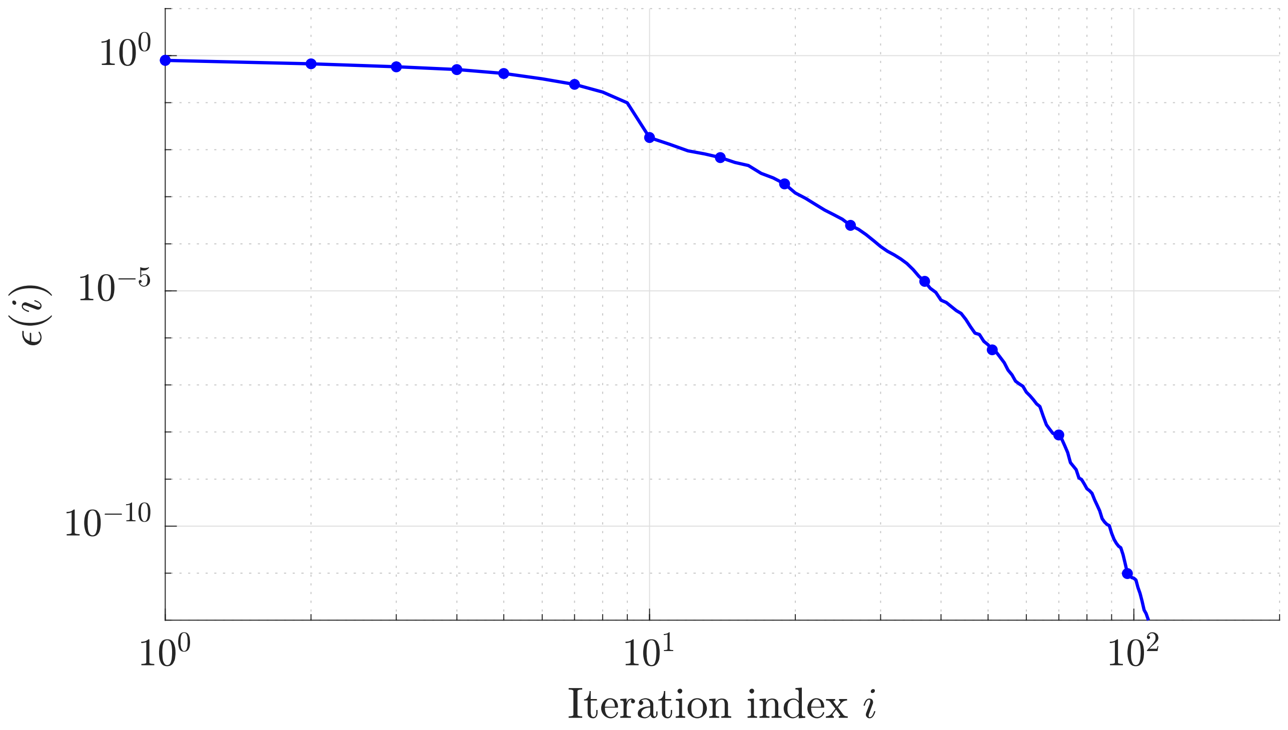
Trace ratio optimization (TRO)

(stationary setting)

$$\underset{X}{\text{maximize}} \; \frac{\mathbb{E}[||X^T\mathbf{y}(t)||^2]}{\mathbb{E}[||X^T\mathbf{v}(t)||^2]}$$

$$\text{subject to } X^TX = I_Q$$

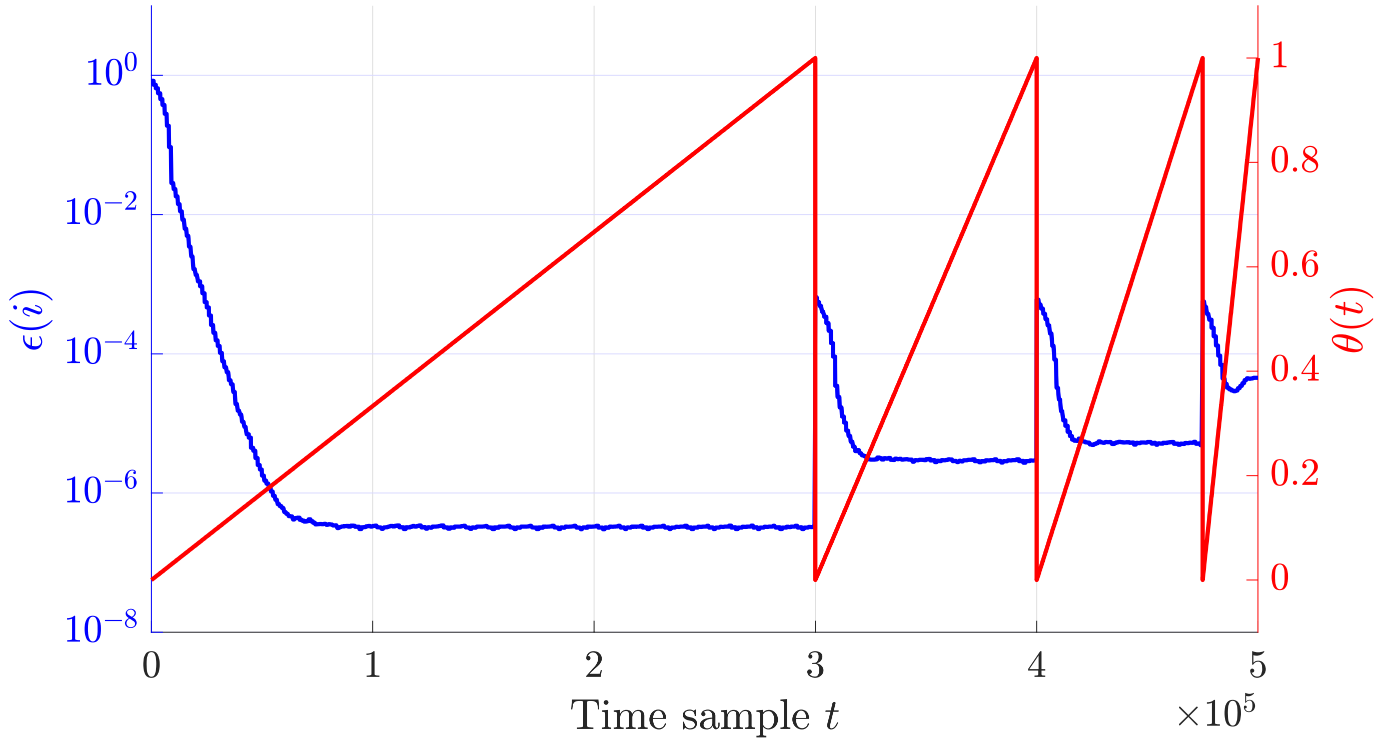
$$\epsilon(i) = \text{median}\left(\frac{||X^i - X^*||_F^2}{||X^*||_F^2}\right)$$



Simulations

TRO (adaptive setting)

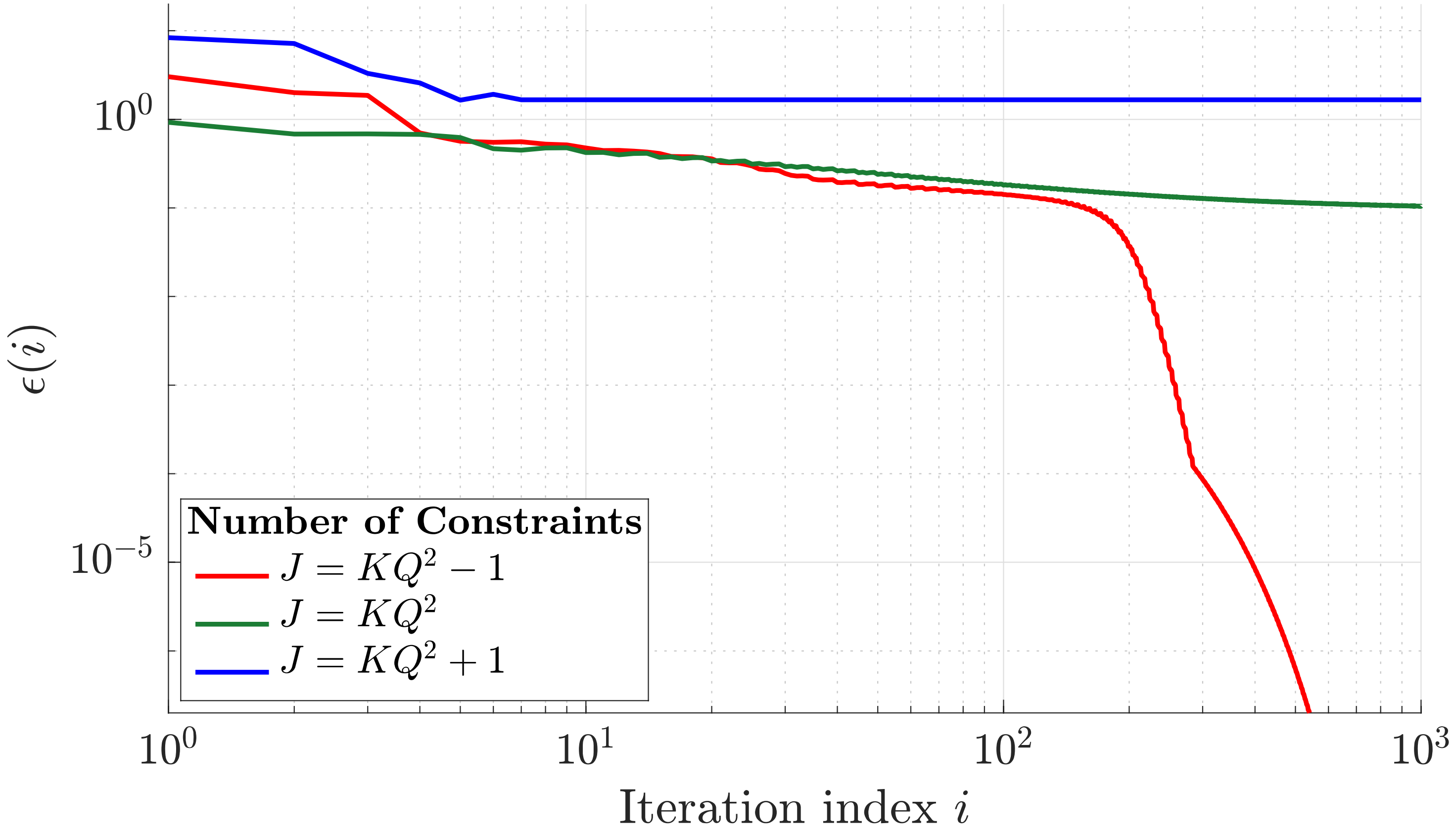
$\theta(t)$ is a time-varying parameter in the generation of the sensor data



Convergence vs number of constraints

LCMV problem (stationary setting)

$$\underset{X}{\text{minimize}}\ \mathbb{E}[||X^T\mathbf{y}(t)||^2]$$
$$\text{subject to } X^T B = H$$

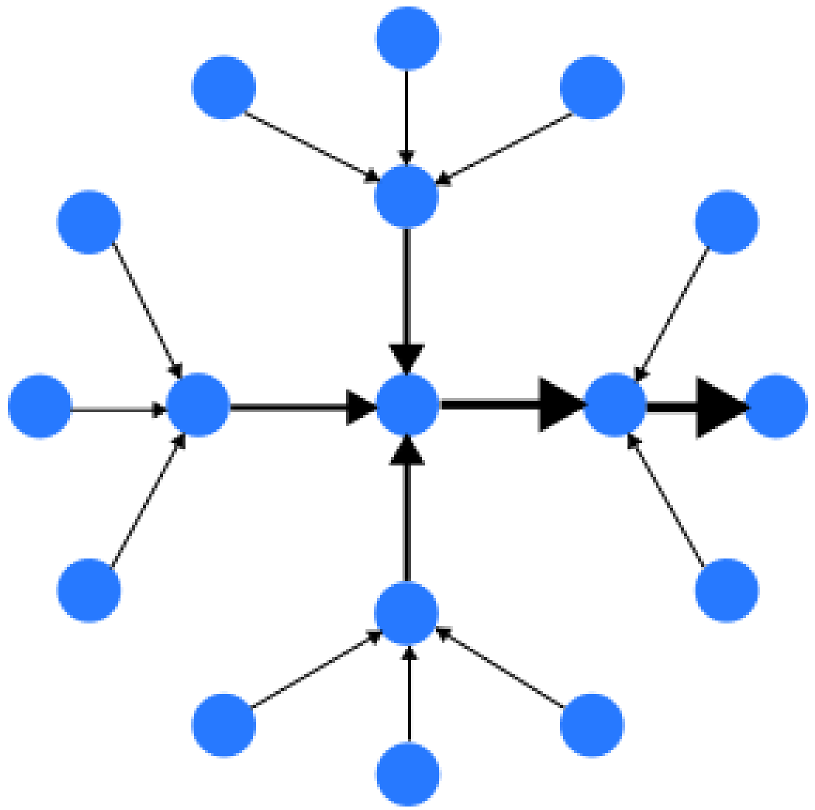


IV- Topology-independent DASF (TI-DASF)

Extending DASF to Arbitrary Topologies
Naïve Approach: Data Relaying

What if we emulate an FC topology by relaying the data?

- Requires **an additional networking layer**
- Worst-case bandwidth (Line topology): $\mathcal{O}(K)$
  - Scales **poorly** with the number of nodes
  - Should ideally be **independent** of the network's size

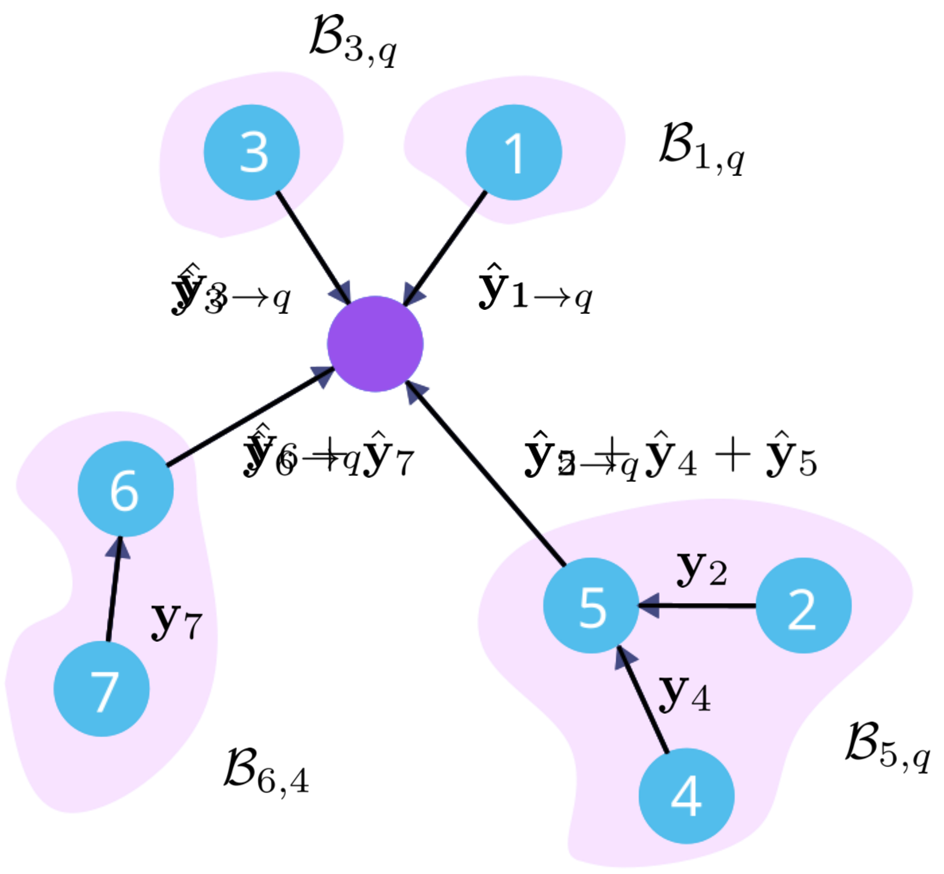


Tree-Topologies: Fuse and Forward

1. **Group** the nodes according to the branch $\mathcal{B}_{n,q}$ they belong to
2. **Recursively sum** (a new batch of) the data within each branch

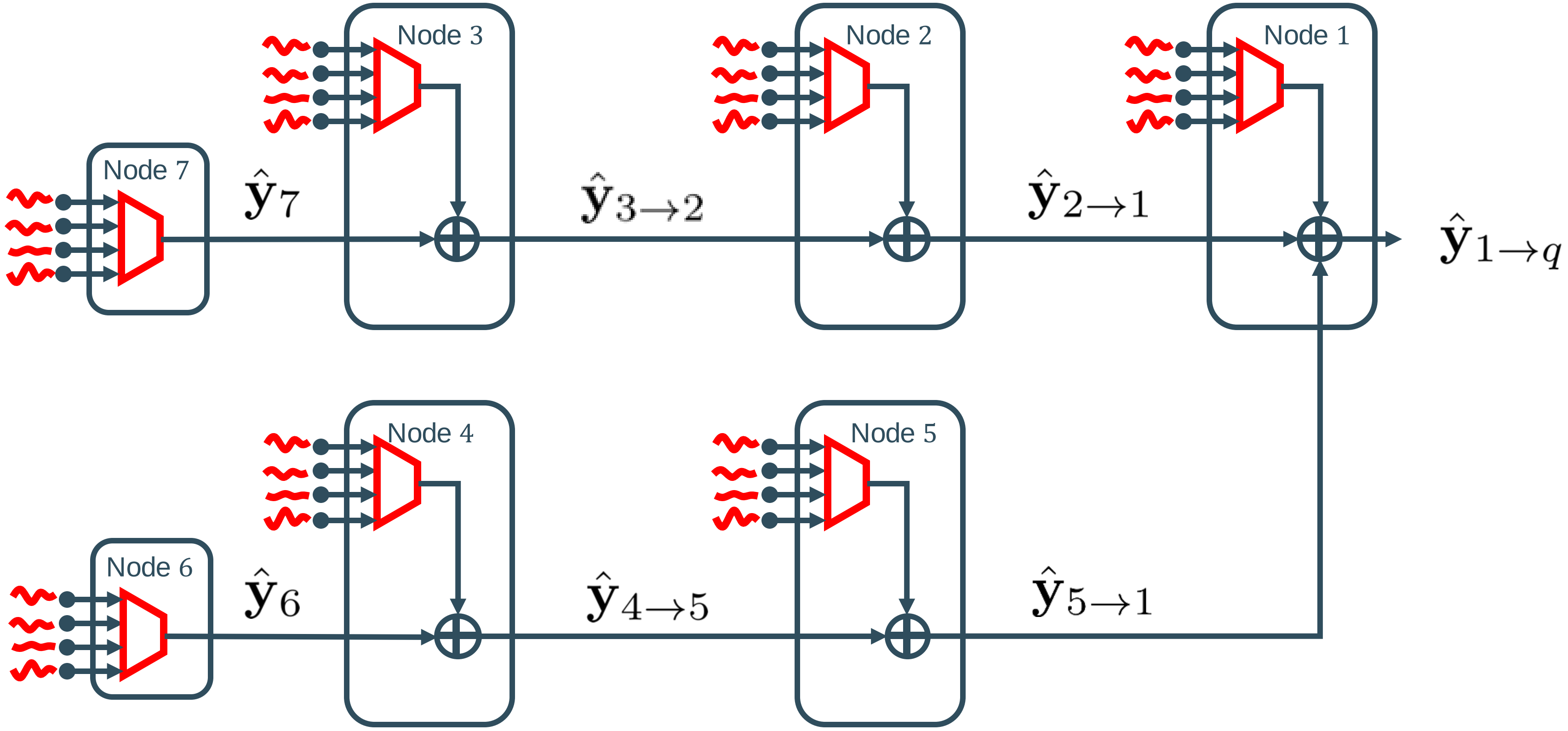
$$\hat{\mathbf{y}}_{n\to q} = \sum_{k\in\mathcal{B}_{nq}} \hat{\mathbf{y}}_k$$

3. **Collect the sums** at the updating node



Tree topology: fuse and forward

Fuse and forward towards updating node q



It is as if the branch consisted of a single node with filter

$$\begin{bmatrix} X_1 \\ X_2 \\ X_3 \\ X_4 \\ X_5 \\ X_6 \\ X_7 \end{bmatrix} \quad \text{and data} \quad \begin{bmatrix} \mathbf{y}_1 \\ \mathbf{y}_2 \\ \mathbf{y}_3 \\ \mathbf{y}_4 \\ \mathbf{y}_5 \\ \mathbf{y}_6 \\ \mathbf{y}_7 \end{bmatrix}$$

With compressed data

$$\hat{\mathbf{y}}_{1\to q}$$

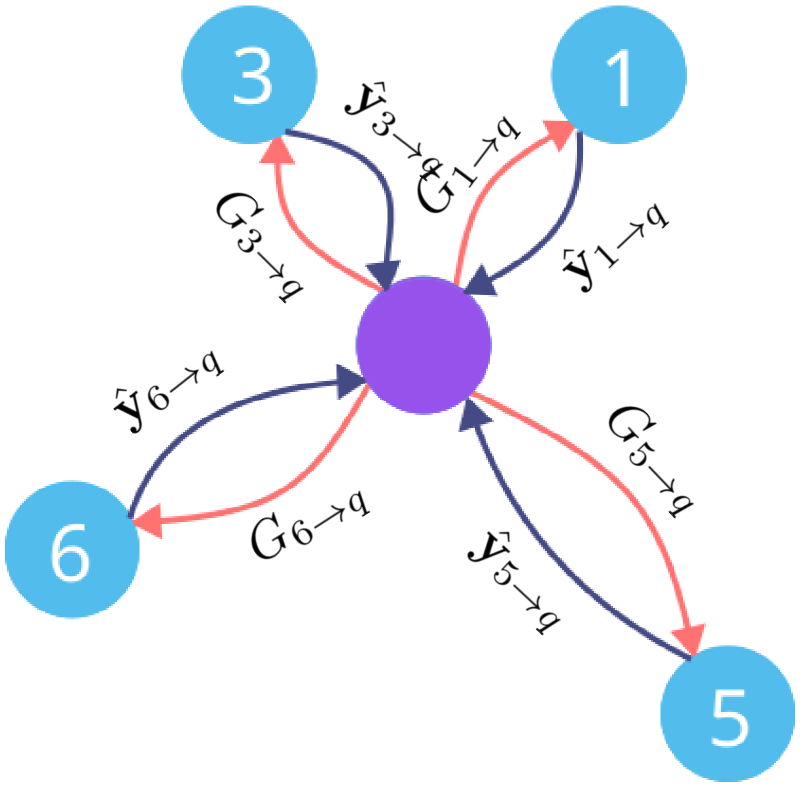
Tree Topology: Updating Node POV

- The updating node has 4 neighbors, sending it 4 signals.
- As if this was a FC network, it constructs

$$\tilde{\mathbf{y}}_q = \begin{bmatrix} \mathbf{y}_q \\ \hat{\mathbf{y}}_{1\to q} \\ \hat{\mathbf{y}}_{3\to q} \\ \hat{\mathbf{y}}_{5\to q} \\ \hat{\mathbf{y}}_{6\to q} \end{bmatrix}$$

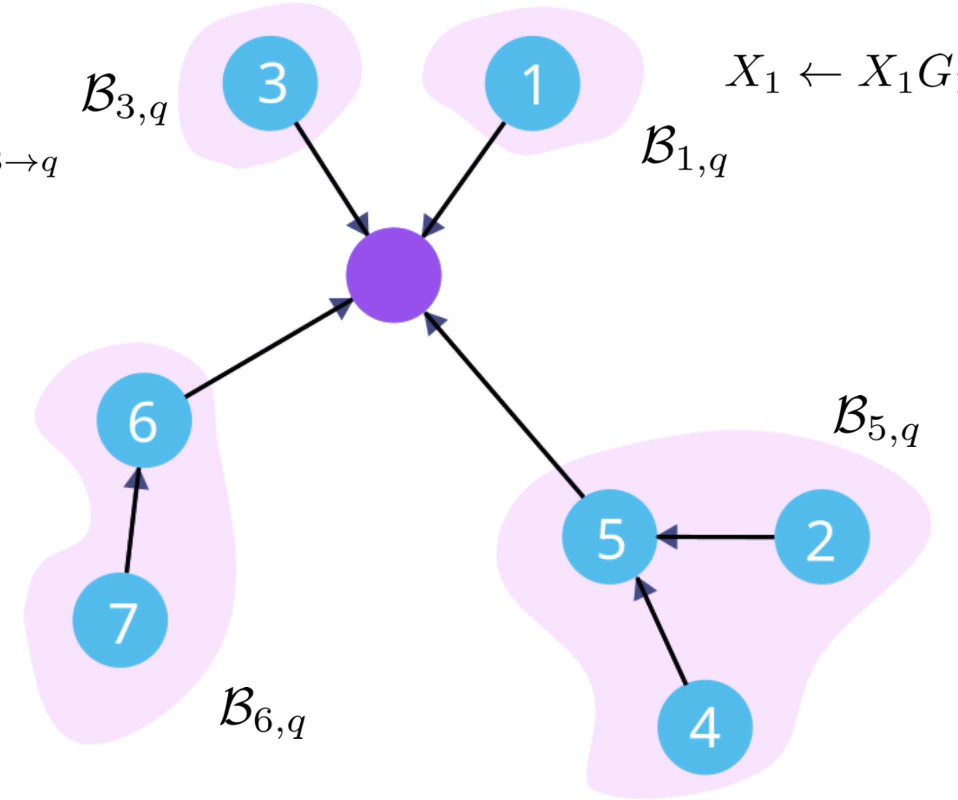
- And solve the usual local problem, obtaining

$$\widetilde{X}_q = \begin{bmatrix} X_q \\ G_{1\to q} \\ G_{3\to q} \\ G_{5\to 4} \\ G_{6\to 4} \end{bmatrix}$$



Tree Topology:
Updating Node POV

- The update is performed in each branch, as if it consisted of a single node i.e.



$$X_1 \leftarrow X_1 G_{1\to q}$$

$$X_3 \leftarrow X_3 G_{3\to q}$$

$$\boxed{X_k^{i+1} \leftarrow X_k^i G_{n\to q} \quad \forall k \in \mathcal{B}_{n,q}}$$

$$\begin{bmatrix} X_6 \\ X_7 \end{bmatrix} \leftarrow \begin{bmatrix} X_6 \\ X_7 \end{bmatrix} G_{6\to q}$$

$$\begin{bmatrix} X_2 \\ X_4 \\ X_5 \end{bmatrix} \leftarrow \begin{bmatrix} X_2 \\ X_4 \\ X_5 \end{bmatrix} G_{5\to q}$$

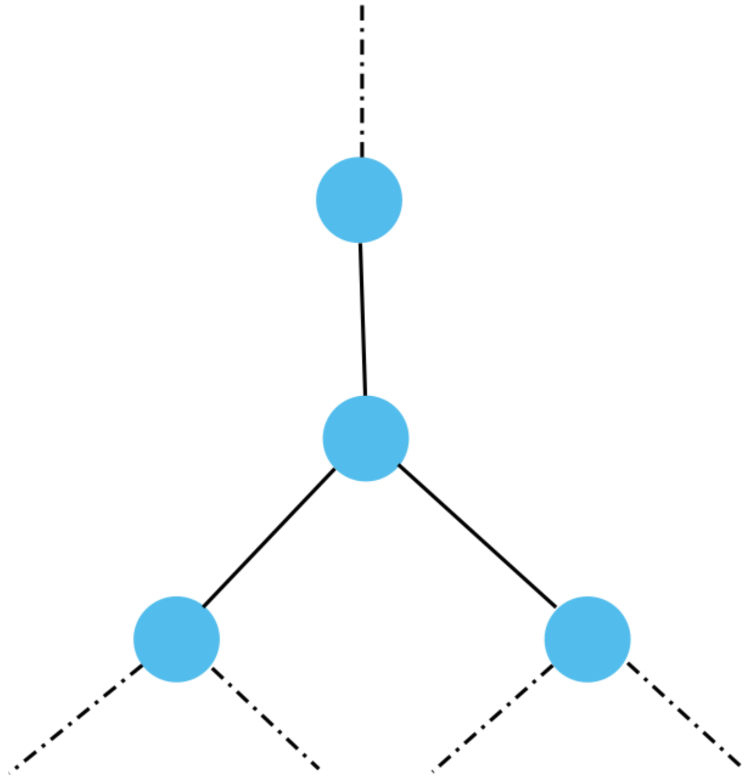
Tree-Topology: Non-Updating Node POV

- Any node k fuses and sends data to remaining neighbors once it has received data from $|\mathcal{N}_k| - 1$ neighbors

  → **data flow arises naturally**

- Updates filter as soon as an update matrix is received

- Forwards the update to other neighbors



Tree topology: fuse and forward

A node is unaware of the existence of other nodes in the network (except its neighbors)

- Each node transmits Q-channel signals (*once* per block of N samples)

- Communication load at each node only depends on # neighbors

  → **Independent of network size**

  → **Fully scalable**

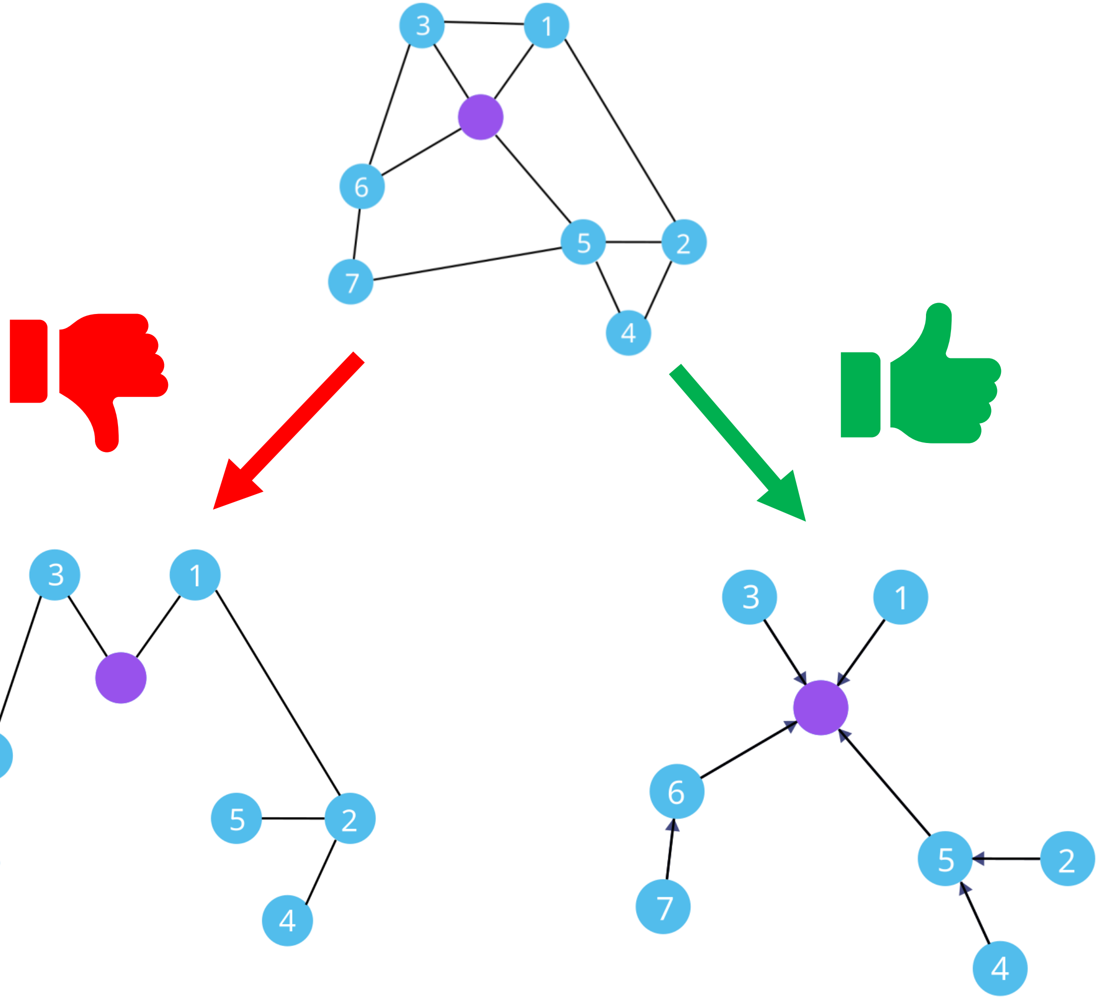
What about arbitrary topologies?

- Prune to spanning tree at each iteration

- **Keep as many neighbors of the updating node as possible**

  → maximize the degrees of freedom in $\tilde{X}_q$

- Pruning can be reused across cycles



Building a Spanning Tree

1. The updating node sends and **ADOPT** to its neighbors
2. Upon receiving an **ADOPT** the first time, the parent link is registered
3. Once they have a **parent**, the nodes send an **ADOPT** to their children

- Tree **expands** from the updating node
- Each node has a **single** parent (first **ADOPT** received)
- **Asynchronous**
- **Decentralized**
- Per-node message complexity $\mathcal{O}(|\mathcal{N}_k|)$
- Time complexity $\mathcal{O}(D)$

procedure `buildtree(`q`)`
- **for** $l \in \mathcal{K}$ **do**
  - Initialize set of children of node l: $\mathcal{L}_l \leftarrow \{\}$
  - Initialize number of acknowledgment messages received: $r_l \leftarrow 0$
- **At node** $l \in \mathcal{N}_q$
  - Set parent to node q: $p_l \leftarrow q$
- **At node** $l \neq q$
  - **while** p_l *is not set* **do**
    - Wait for message (ADOPT, m)
    - Set parent to node m: $p_l \leftarrow m$
    - Send (OK, l) to p_l as an acknowledgment
  - Send (ADOPT, l) to $\mathcal{N}_l \setminus \{p_l\}$
  - **while** $r_l < |\mathcal{N}_l|$ **do**
    - Wait for message (a, m)
    - **if** $a = \mathit{ADOPT}$ **then**
      - Send $(\text{NOK}, \cdot)$ to node m to indicate that it already has a parent
    - **else**
      - **if** $a = \mathit{OK}$ **then**
        - Add m to set of children of l: $\mathcal{L}_l \leftarrow \mathcal{L}_l + \{m\}$
    - $r_l \leftarrow r_l + 1$

Per-Node Bandwidth and Complexity

	Centralized	Fully-Connected	Arbitrary
TX Bandwidth	$\mathcal{O}(K)$	$\mathcal{O}(Q)$	$\mathcal{O}(Q)$
RX Bandwidth	$\mathcal{O}(K)$	$\mathcal{O}(KQ)$	$\mathcal{O}(\lvert\mathcal{N}_k\rvert Q)$
Problem dimension	$\sum_k^K M_k$	$(K-1)Q^2 + M_k Q$	$\lvert\mathcal{N}_k\rvert Q^2 + M_k Q$

Independent of network size K

Convergence

- Same convergence properties under LICQ-like conditions
- These LICQ-like conditions result in a new bound on the maximum # constraints

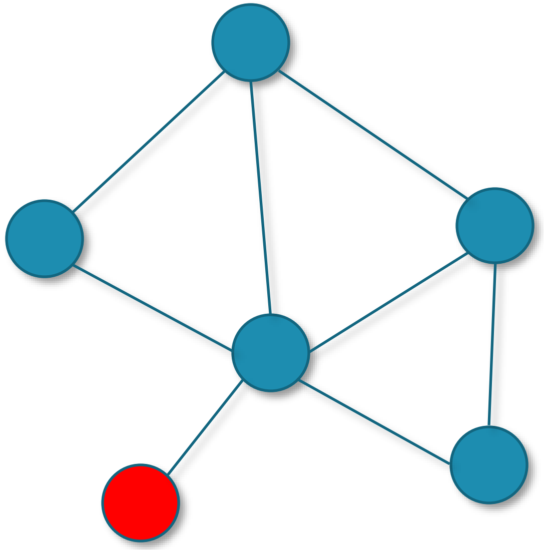
$$\boxed{J \leq \min\left(\frac{Q^2}{K-1}\sum_{k\in\mathcal{K}}|\mathcal{N}_k|,\ (1+\min_{k\in\mathcal{K}}|\mathcal{N}_k|)Q^2\right)}$$

K : Number of nodes in the network
Q : Number of columns of X
$\mathcal{N}_k$: Neighbors of node k (excl. node k itself)

Convergence

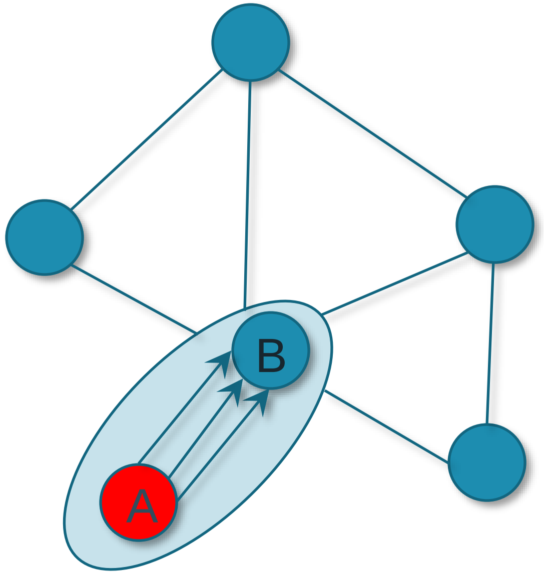
One single node with few neighbors can have a large impact

$$J \leq \min\left(\frac{Q^2}{K-1}\sum_{k\in\mathcal{K}}|\mathcal{N}_k|,\ \left(1+\boxed{\min_{k\in\mathcal{K}}|\mathcal{N}_k|}\right)Q^2\right)$$



If more constraints needed
→ merge nodes

i.e., A sends raw data to B, who treats this as own sensor data.



Examples

Stiefel manifold constraints: $X^T X = I_Q$

$$X \text{ is } M \times Q \qquad \rightarrow J = \frac{Q(Q+1)}{2}$$

always satisfied

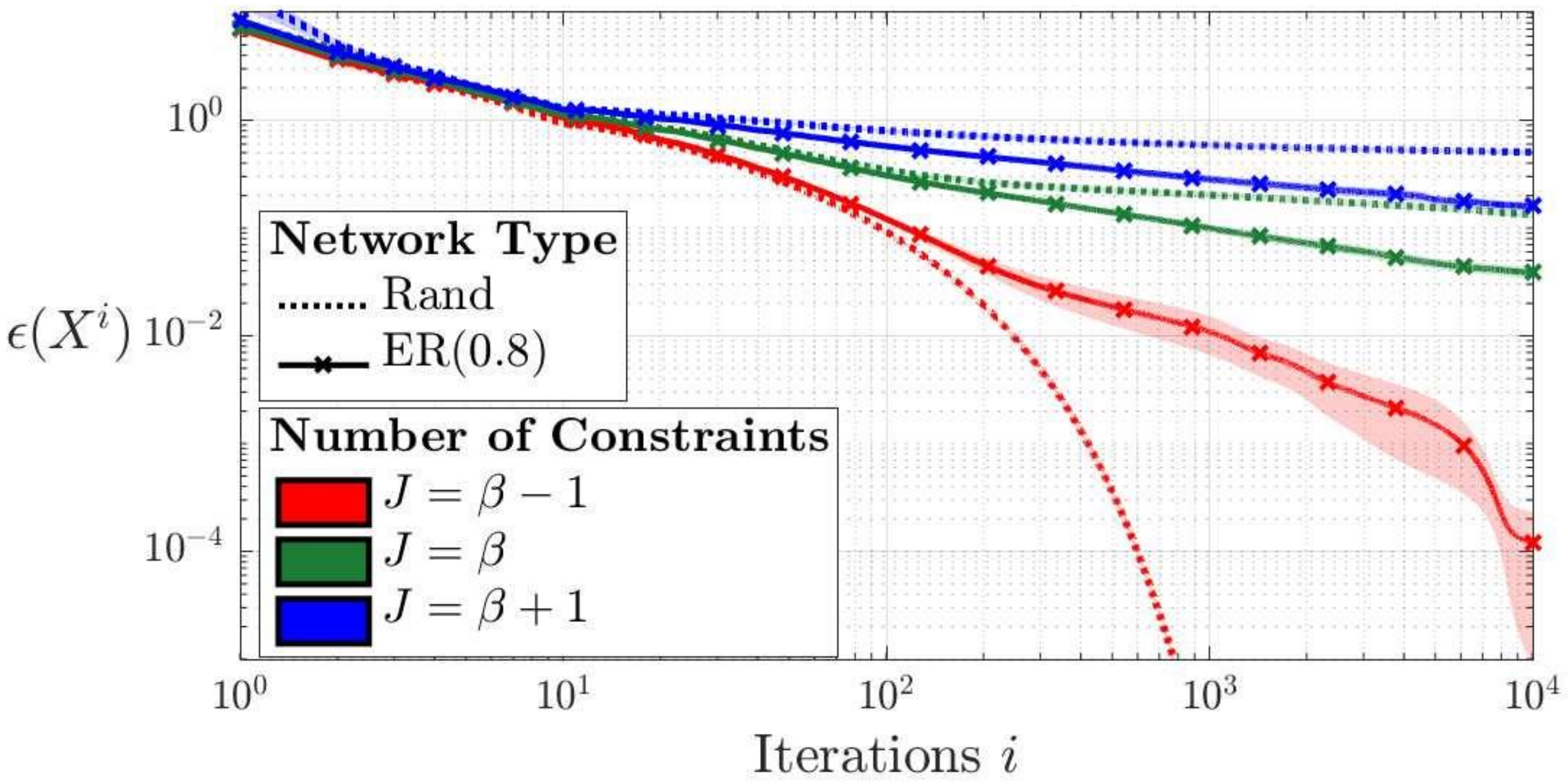
Linear constraints: $X^T B = H$

$$B \text{ is } M \times L \qquad \rightarrow J = QL$$

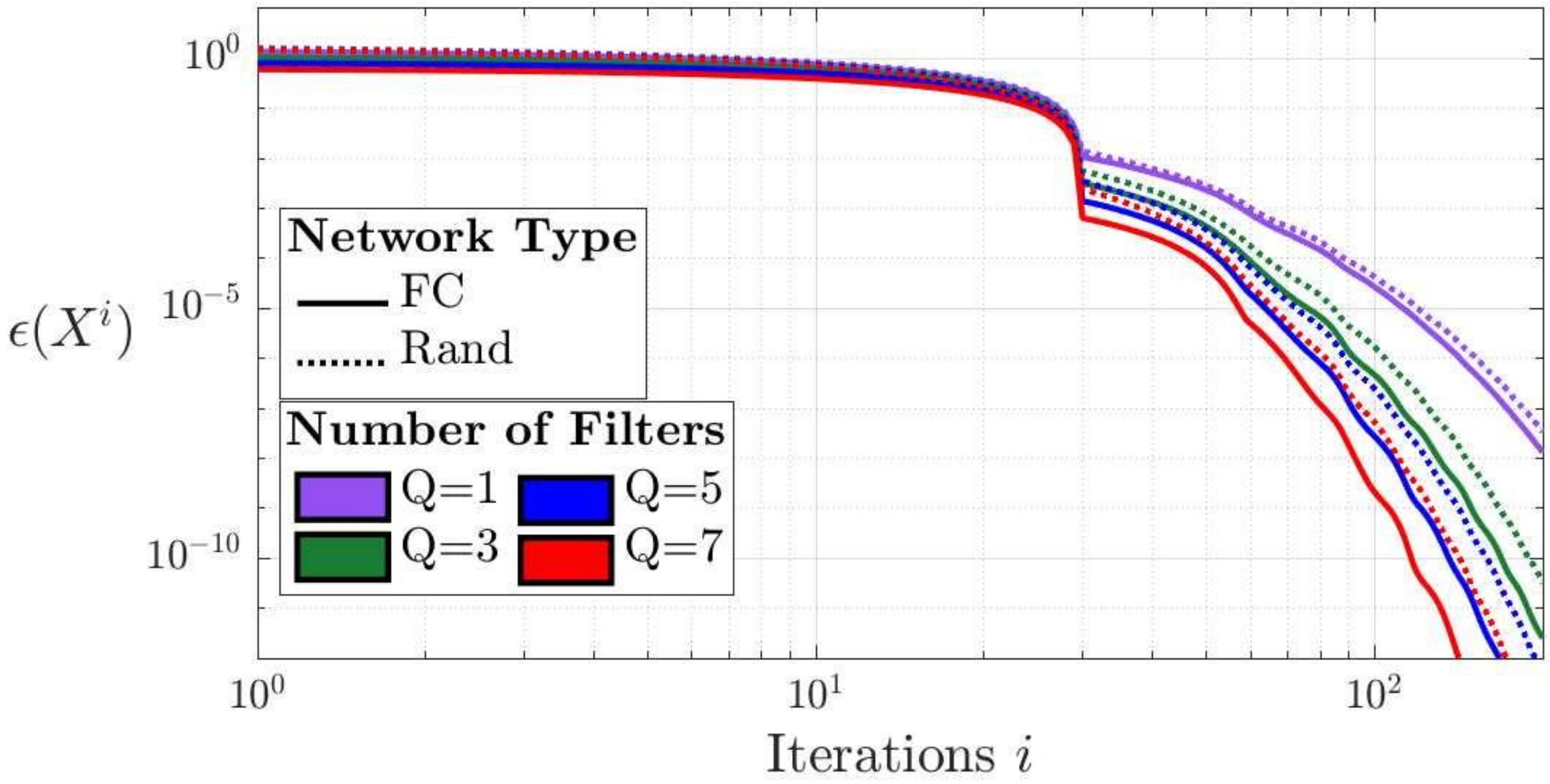
$$L \leq (1 + \min |\mathcal{N}_q|)Q$$

$$L \leq \frac{Q}{K-1} \sum |\mathcal{N}_q|$$

Convergence
Impact of Constraints Bounds



Convergence
Impact of Bandwidth



V- The DASF toolbox

DASF Toolbox

Purpose:

- Experiment with the DASF algorithm on your favorite spatial filter

Entry-Point:

A function expecting the following inputs:

- Function containing your **solver** for the centralized spatial filter design (i.e., function)
- Input signals
- Network topology
- Sequence of updating nodes
- Number of channels and output filters

(*)

(*) can also be automatically generated

Several examples available:

- PCA/GEVD
- MMSE
- TRO
- QCQP
- SCQP
- LCMV
- CCA
- **Easily add your own!**

Available for both MATLAB and Python

http://tinyurl.com/DASFToolbox

Scan me

VI- Extensions

Non-smooth problems

- Convergence and optimality has also been shown for **non-smooth problems**

$$\begin{aligned} \underset{X}{\text{minimize}}\ & f(X^T\mathbf{y}(t)) + g(X^T\mathbf{\Gamma}) \\ \text{subject to}\ & l_j(X_k^T\mathbf{y}_k(t)) = 0, \\ & h_j(X_k^T\mathbf{y}_k(t)) \leq 0 \end{aligned}$$

- **Non-smooth term** must be **node-separable**, i.e. $g(X^T\mathbf{\Gamma}) = \sum_k g(X_k^T\mathbf{\Gamma}_k)$

(satisfied for l_1, l_2, $l_{1,2}$, Frobenius norm, …)

- Limitations:
- **constraints** must also be **node-separable**
  - In arbitrary networks, constraints can depend **on** B **only** , not $\mathbf{y}(t)$

Non-smooth problems
Examples

- Block-sparse generalized CCA (SUMCORR variant):

$$\min_X \mathbb{E}[\mathrm{tr}(X^T\mathbf{y}(t)\mathbf{y}(t)^T X)] + \sum_k ||X_k||_{2,1}$$
$$\text{s.t.}\quad \mathbb{E}[X_k^T\mathbf{y}_k(t)\mathbf{y}_k(t)^T X_k] = I_Q \quad \forall k \in \mathcal{K}$$

- Block-sparse regularized MMSE / MWF

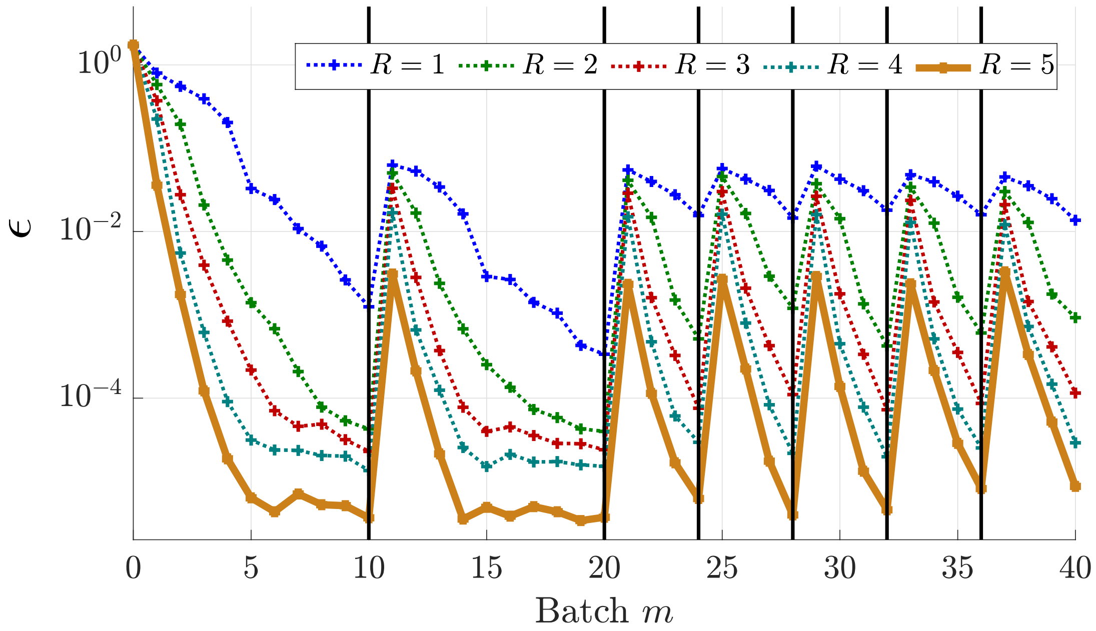
$$\min_X \mathbb{E}[||X^T\mathbf{y}(t) - \mathbf{d}(t)||_F^2] + \sum_k ||X_k||_F$$
$$\text{s.t.}\quad \mathbb{E}[||X_k^T\mathbf{y}_k(t)||_F^2] \leq P_k \quad \forall k \in \mathcal{K}$$

Tracking improvements

- Re-use data from previous iterations

- Efficient method allows an R times faster convergence with an L-fold increase in bandwidth with $R \gg L$

  (typically, $R = K, \;\; L = 2$)



Parallel updates

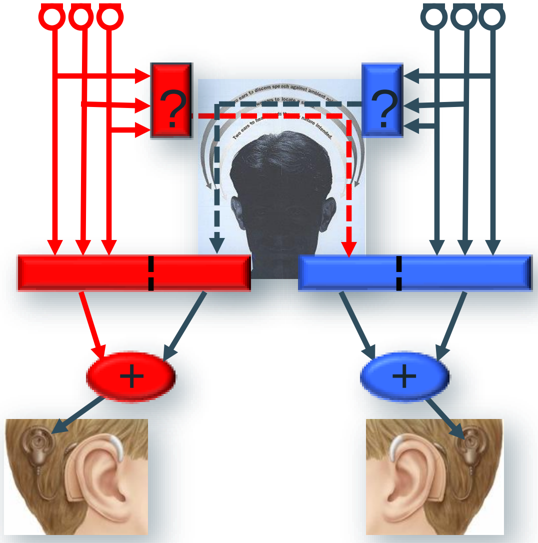
- Nodes (or subsets of nodes) can update simultaneously
- Requires relaxation on the updates: $X^{i+1} \leftarrow \alpha X^{i} + (1-\alpha)X^{i+1}$
- No analytical convergence results (yet)

Node-specific problems

Problem	Objective to minimize at node k	Constraints
MMSE	$\min \mathbb{E}[\|\|\mathbf{d}_k(t) - X(k)^T\mathbf{y}(t)\|\|^2]$	—
LCMV	$\min \mathbb{E}[\|\|X(k)^T\mathbf{y}(t)\|\|^2]$	$X(k)^T B = H_k$

- Each node wishes to solve a **node-specific** problem

$$\begin{aligned} \underset{X(k)\in\mathbb{R}^{M\times Q}}{\text{minimize}} \quad & f_k\left(X(k)^T\mathbf{y}(t), X(k)^T B\right) \\ \text{subject to} \quad & X(k) \in \mathcal{S}_k \end{aligned}$$



- Works if the node-specific solutions are the same up to a full rank transformation(*):

$$\forall(k,l),\ \exists D_{k,l} :\ X(k)^* = X(l)^* \cdot D_{k,l}$$

(*) this happens in, e.g., speech denoising, where each node uses a different reference microphone to estimate the same speech signal *[Doclo et al. 2009]*

Fractional programming

Problem	Objective to minimize	Constraints
TRO	$-\frac{\mathbb{E}[\|\|X^T\mathbf{y}(t)\|\|^2]}{\mathbb{E}[\|\|X^T\mathbf{v}(t)\|\|^2]}$	$X^TX = I_Q$
RTLS	$\frac{\mathbb{E}[\|\|\mathbf{x}^T\mathbf{y}(t)-d(t)\|\|^2]}{1+\|\|\mathbf{x}\|\|^2}$	$\|\|\mathbf{x}^TL\|\|^2 \leq \delta^2$

- Special version of DASF for fractional programs:

$$\min_X \frac{f_1(X)}{f_2(X)}, \text{ s.t., } X \in \mathcal{S}$$

- Fractional programs are solved with an iterative procedure (Dinkelbach)

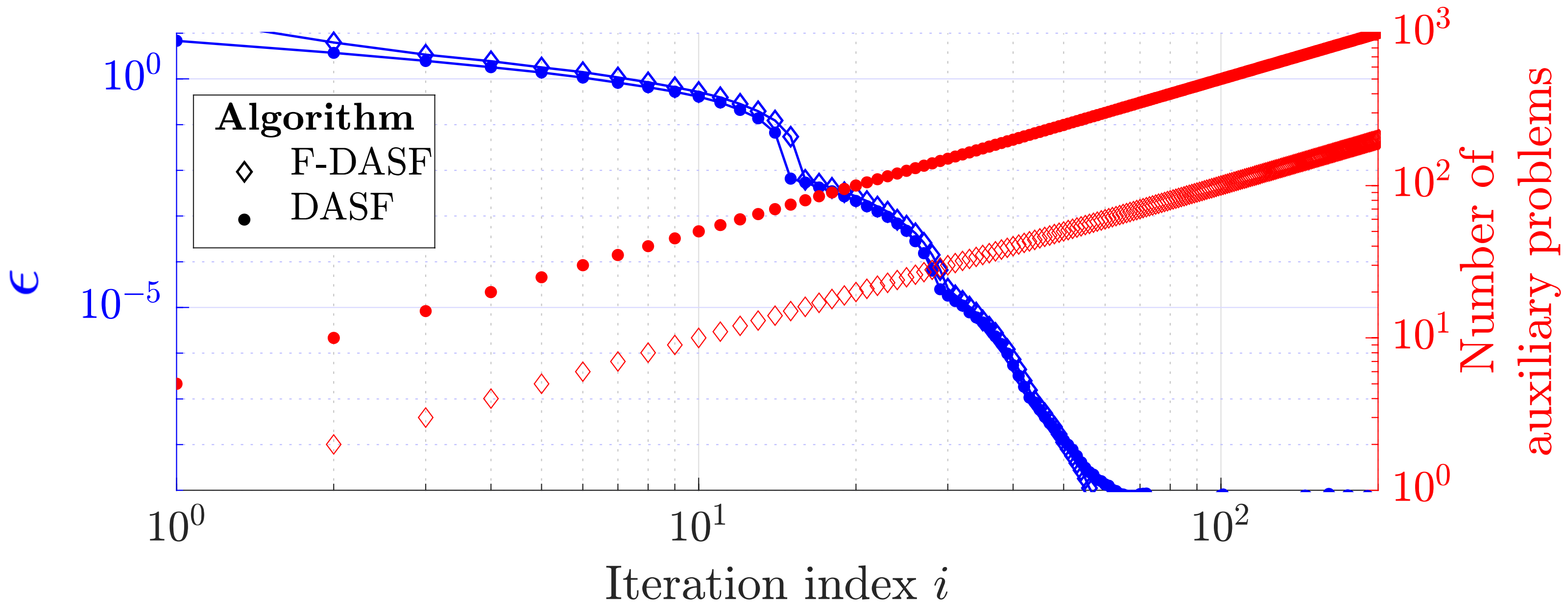
  → large computational cost at each node

- Main idea: interleave steps of DASF with iterative solver

  → with some modifications in the proof, can still prove convergence

Fractional programming

- Convergence rate similar to DASF, with signifcantly less computations



VII- Conclusion

Conclusions

- Generic algorithmic framework for **distributed** & **adaptive** spatial filtering
- Covers large number of optimization problems including PCA, GEVD, CCA, MMSE, LCMV, TRO, MAXVAR, SUMCORR, sparsity norms, etc.
- **Convergence guarantees** under mild assumptions generally satisfied in practice
- Various extensions

Bibliography: key papers

[Musluoglu et al. 2023a] C. A. Musluoglu and A. Bertrand, "A Unified Algorithmic Framework for Distributed Adaptive Signal and Feature Fusion Problems—Part I: Algorithm Derivation," in *IEEE Transactions on Signal Processing*, vol. 71, pp. 1863-1878, 2023

[Musluoglu et al. 2023b] C. A. Musluoglu, C. Hovine and A. Bertrand, "A Unified Algorithmic Framework for Distributed Adaptive Signal and Feature Fusion Problems — Part II: Convergence Properties," in *IEEE Transactions on Signal Processing*, vol. 71, pp. 1879-1894, 2023

Bibliography (I)

[Doclo et al. 2009] S. Doclo, M. Moonen, T. Van den Bogaert and J. Wouters, "Reduced-Bandwidth and Distributed MWF-Based Noise Reduction Algorithms for Binaural Hearing Aids," in IEEE Trans. Audio, Speech, and Language Processing, vol. 17, no. 1, pp. 38-51, 2009.

[Musluoglu et al. 2023a] C. A. Musluoglu and A. Bertrand, "A Unified Algorithmic Framework for Distributed Adaptive Signal and Feature Fusion Problems—Part I: Algorithm Derivation," in *IEEE Trans. Signal Processing*, vol. 71, pp. 1863-1878, 2023

[Musluoglu et al. 2023b] C. A. Musluoglu, C. Hovine and A. Bertrand, "A Unified Algorithmic Framework for Distributed Adaptive Signal and Feature Fusion Problems — Part II: Convergence Properties," in *IEEE Trans. Signal Processing*, vol. 71, pp. 1879-1894, 2023

[Bertrand et al. 2011] A. Bertrand, and M. Moonen, "Distributed LCMV beamforming in wireless sensor networks with node-specific desired signals." *2011 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2011.

[Zeng et al. 2013] Y. Zeng and Richard C. Hendriks. "Distributed delay and sum beamformer for speech enhancement via randomized gossip." *IEEE/ACM Transactions on Audio, Speech, and Language Processing, vol. 22, no. 1* 22.1, pp. 260-273, 2013.

Bibliography (II)

[Li et al. 2011] L. Li, A. Scaglione, and J. H. Manton. "Distributed principal subspace estimation in wireless sensor networks." IEEE Journal of Selected Topics in Signal Processing vol. 5, no.4, pp725-738, 2011.

[Mota et al. 2011] J.F.C Mota, et al. "Distributed basis pursuit." IEEE Transactions on Signal Processing vol. 60, pp. 1942-1956, 2011.

[Markovich et al. 2015] S. Markovich-Golan, et al. "Optimal distributed minimum-variance beamforming approaches for speech enhancement in wireless acoustic sensor networks." Signal Processing, vol.107, pp. 4-20, 2015.

[Fu et al. 2016] X. Fu, et al. "Efficient and distributed algorithms for large-scale generalized canonical correlations analysis." 2016 IEEE 16th international conference on data mining (ICDM). IEEE, 2016.

[Bertrand et al. 2018] A. Bertrand, and M. Moonen. "Distributed canonical correlation analysis in wireless sensor networks with application to distributed blind source separation." IEEE Transactions on Signal Processing vol. 63, no.18, pp. 4800-4813, 2018.

[Zhang et al. 2019] J. Zhang, et al. "Distributed rate-constrained LCMV beamforming." IEEE Signal Processing Letters, vol. 26, no. 5, pp. 675-679, 2019.

[Musluoglu et al. 2021] C.A. Musluoglu,, and A. Bertrand. "Distributed adaptive trace ratio optimization in wireless sensor networks." IEEE Transactions on Signal Processing, vol. 69 pp. 3653-3670, 2021.