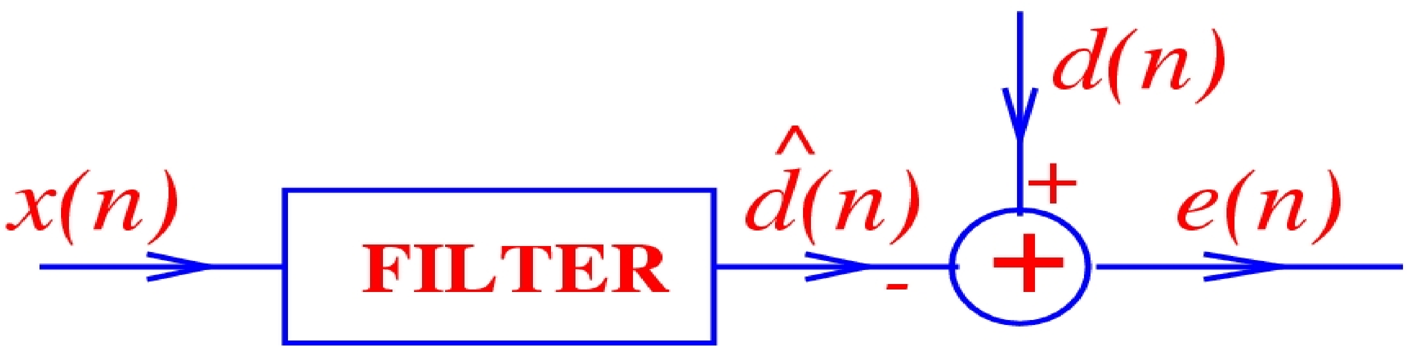
Adaptive Filtering

Recall optimal filtering: Given

$$x(n) = d(n) + v(n),$$

estimate and extract $d(n)$ from the current and past values of $x(n)$.



Let the filter coefficients be

$$\boldsymbol{w} = \begin{bmatrix} w_0 \\ w_1 \\ \vdots \\ w_{N-1} \end{bmatrix}.$$

Filter output:

$$y(n) = \sum_{k=0}^{N-1} w_k^* x(n-k) = \boldsymbol{w}^H \boldsymbol{x}(n) = \widehat{d}(n),$$

where

$$\boldsymbol{x}(n) = \begin{bmatrix} x(n) \\ x(n-1) \\ \vdots \\ x(n-N+1) \end{bmatrix}.$$

Wiener-Hopf equation:

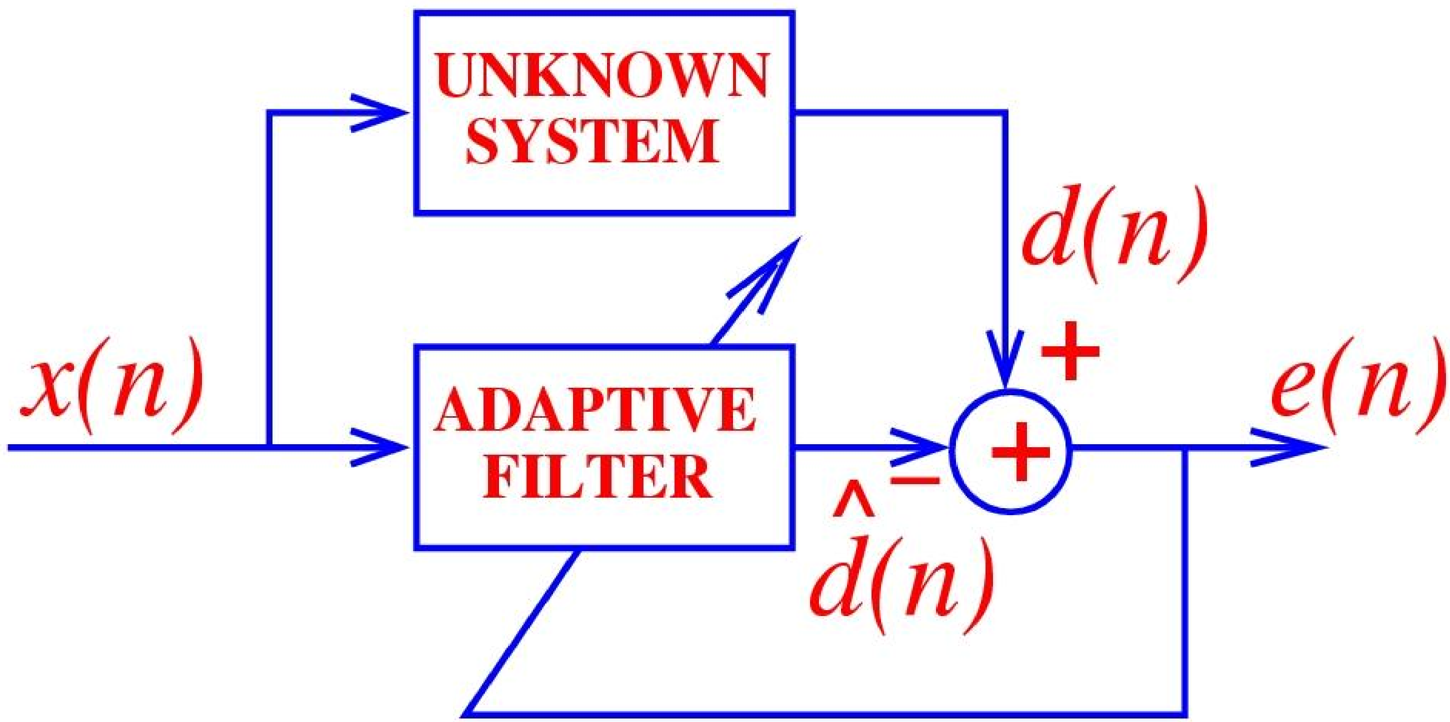
$$R(n)\boldsymbol{w}(n) = \boldsymbol{r}(n) \quad \longrightarrow \quad \boldsymbol{w}_{\text{opt}}(n) = R(n)^{-1}\boldsymbol{r}(n),$$

where

$$\begin{aligned} R(n) &= \mathrm{E}\,\{\boldsymbol{x}(n)\boldsymbol{x}(n)^H\}, \\ \boldsymbol{r}(n) &= \mathrm{E}\,\{\boldsymbol{x}(n)d(n)^*\}. \end{aligned}$$

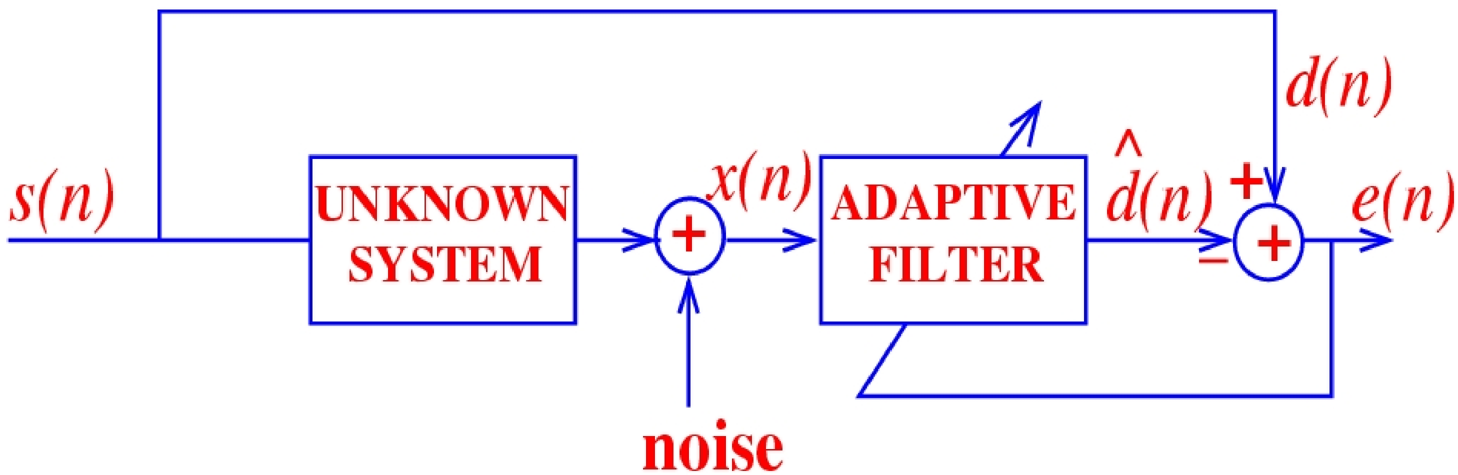
Adaptive Filtering (cont.)

Example 1: Unknown system identification.



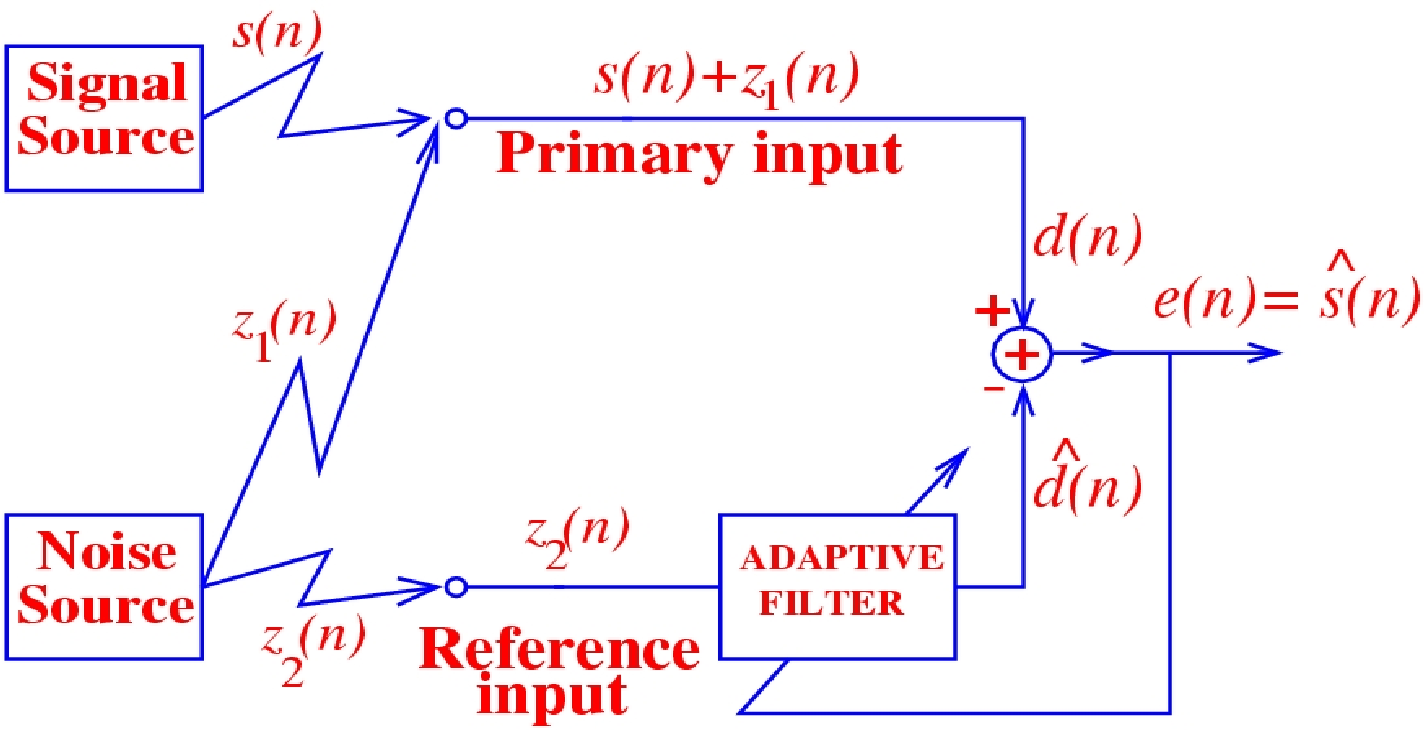
Adaptive Filtering (cont.)

Example 2: Unknown system equalization.



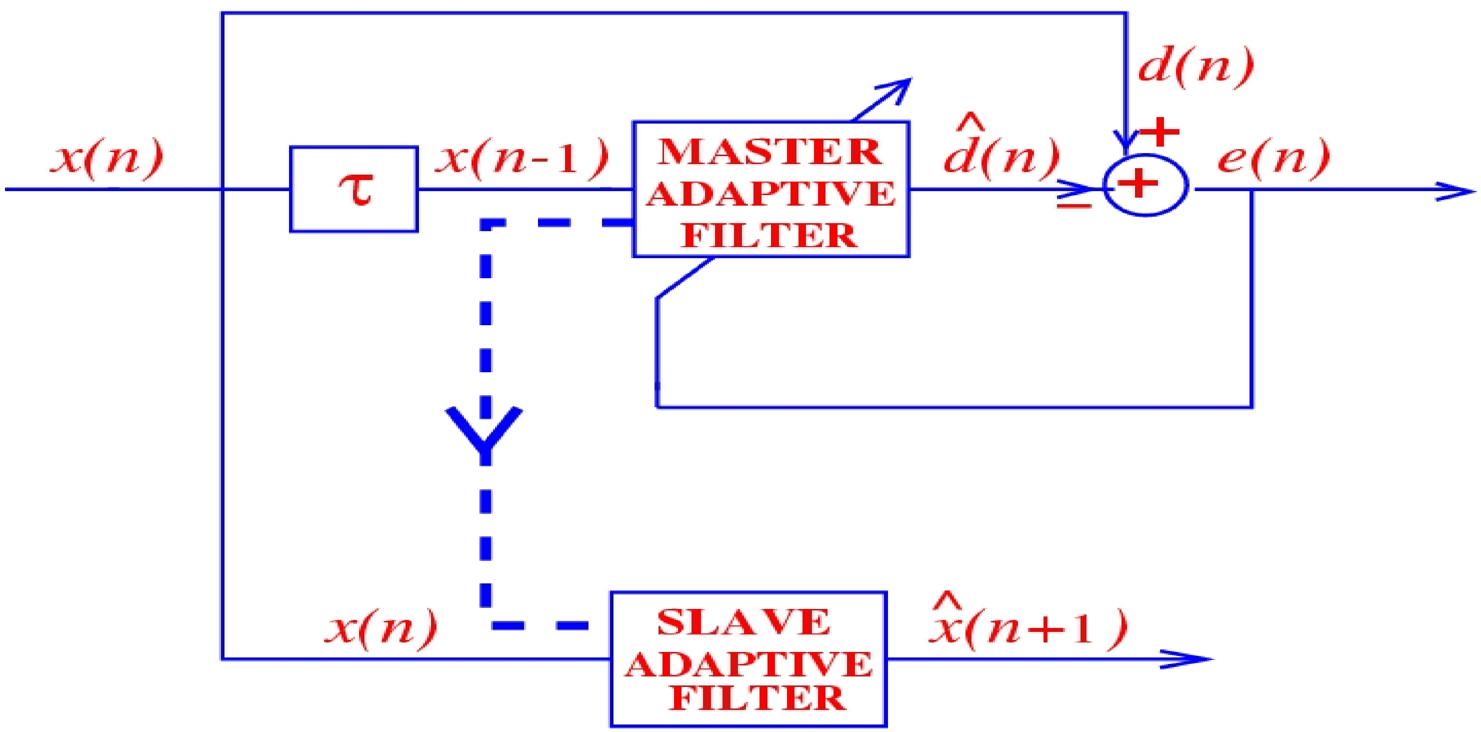
Adaptive Filtering (cont.)

Example 3: Noise cancellation.



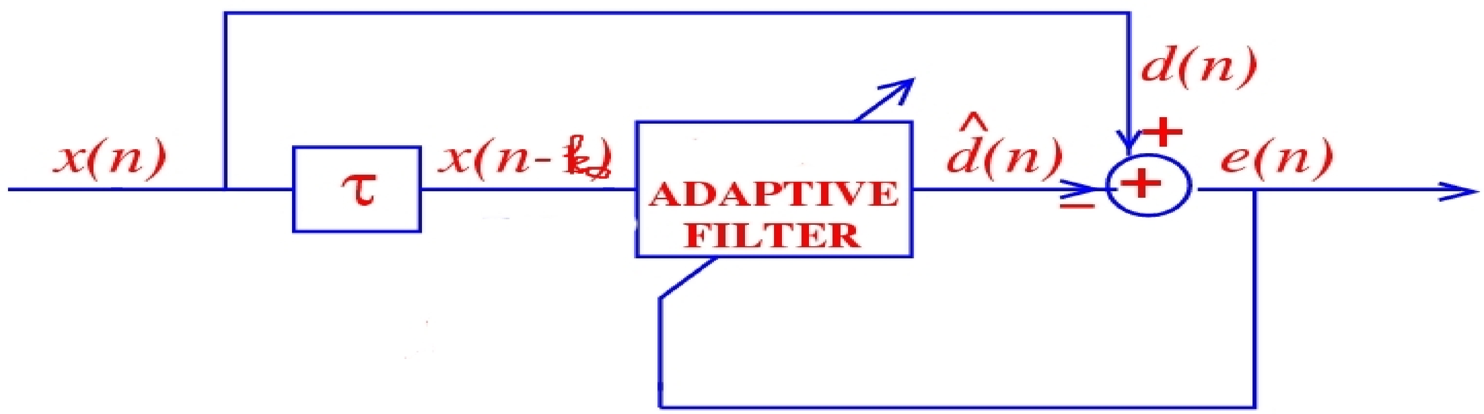
Adaptive Filtering (cont.)

Example 4: Signal linear prediction.



Adaptive Filtering (cont.)

Example 5: Interference cancellation without reference input.



Adaptive Filtering (cont.)

Idea of the *Least-Mean-Square (LMS) algorithm*:

$$\boldsymbol{w}_{k+1} = \boldsymbol{w}_k - \mu(\nabla_{\boldsymbol{w}} \mathrm{E}\{|e_k|^2\})^*, \qquad (*)$$

where the indices are given as subscripts [e.g. $d(k) = d_k$], and

$$\begin{aligned} \mathrm{E}\{|e_k|^2\} &= \mathrm{E}\{|d_k - \boldsymbol{w}_k^H \boldsymbol{x}_k|^2\} \\ &= \mathrm{E}\{|d_k|^2\} - \boldsymbol{w}_k^H \boldsymbol{r} - \boldsymbol{r}^H \boldsymbol{w}_k + \boldsymbol{w}_k^H R \boldsymbol{w}_k, \\ (\nabla_{\boldsymbol{w}} \mathrm{E}\{|e_k|^2\})^* &= R\boldsymbol{w} - \boldsymbol{r}. \end{aligned}$$

Use single-sample estimates of R and $\boldsymbol{r}$:

$$\widehat{R} = \boldsymbol{x}_k \boldsymbol{x}_k^H, \quad \widehat{\boldsymbol{r}} = \boldsymbol{x}_k d_k^*,$$

and insert them into $(*)$:

$$\boldsymbol{w}_{k+1} = \boldsymbol{w}_k + \mu \boldsymbol{x}_k e_k^*, \quad e_k = d_k - \boldsymbol{w}_k^H \boldsymbol{x}_k \quad \leftarrow \text{LMS alg.}$$

Adaptive Filtering: Convergence Analysis

Convergence analysis: Subtract $\boldsymbol{w}_{\text{opt}}$ from both sides of the previous equation:

$$\underbrace{\boldsymbol{w}_{k+1} - \boldsymbol{w}_{\text{opt}}}_{\boldsymbol{v}_{k+1}} = \underbrace{\boldsymbol{w}_k - \boldsymbol{w}_{\text{opt}}}_{\boldsymbol{v}_k} + \mu \boldsymbol{x}_k (d_k^* - \boldsymbol{x}_k^H \boldsymbol{w}_k) \quad (**)$$

and note that

$$\begin{aligned}
\boldsymbol{x}_k (d_k^* - \boldsymbol{x}_k^H \boldsymbol{w}_k) &= \boldsymbol{x}_k d_k^* - \boldsymbol{x}_k \boldsymbol{x}_k^H \boldsymbol{w}_k \\
&= \boldsymbol{x}_k d_k^* - \boldsymbol{x}_k \boldsymbol{x}_k^H \boldsymbol{w}_k + \boldsymbol{x}_k \boldsymbol{x}_k^H \boldsymbol{w}_{\text{opt}} - \boldsymbol{x}_k \boldsymbol{x}_k^H \boldsymbol{w}_{\text{opt}} \\
&= (\boldsymbol{x}_k d_k^* - \boldsymbol{x}_k \boldsymbol{x}_k^H \boldsymbol{w}_{\text{opt}}) - \boldsymbol{x}_k \boldsymbol{x}_k^H \boldsymbol{v}_k.
\end{aligned}$$

Observe that

$$\mathrm{E}\left\{\boldsymbol{x}_k(d_k^* - \boldsymbol{x}_k^H\boldsymbol{w}_k)\right\} = \underbrace{\boldsymbol{r} - R\boldsymbol{w}_{\mathrm{opt}}}_{\boldsymbol{0}} - R\mathrm{E}\{\boldsymbol{v}_k\} = -R\mathrm{E}\{\boldsymbol{v}_k\}.$$

Let $\boldsymbol{c}_k = \mathrm{E}\{\boldsymbol{v}_k\}$. Then

$$\boldsymbol{c}_{k+1} = [I - \mu R]\boldsymbol{c}_k \qquad (***)$$

Sufficient condition for convergence:

$$\|\boldsymbol{c}_{k+1}\| < \|\boldsymbol{c}_k\| \quad \forall k.$$

Adaptive Filtering: Convergence Analysis

Let us premultiply both parts of the equation $(***)$ by the matrix U^H of the eigenvectors of R, where

$$R = U\Lambda U^H.$$

Then, we have

$$\underbrace{U^H \boldsymbol{c}_{k+1}}_{\widehat{\boldsymbol{c}}_{k+1}} = U^H[I - \mu R]\underbrace{UU^H}_{I}\boldsymbol{c}_k,$$

and, hence

$$\widehat{\boldsymbol{c}}_{k+1} = [I - \mu\Lambda]\widehat{\boldsymbol{c}}_k.$$

Since

$$\|\boldsymbol{c}_k\|^2 = \boldsymbol{c}_k^H \boldsymbol{c}_k = \boldsymbol{c}_k^H \underbrace{UU^H}_{I}\boldsymbol{c}_k = \widehat{\boldsymbol{c}}_k^H \widehat{\boldsymbol{c}}_k = \|\widehat{\boldsymbol{c}}_k\|^2,$$

the sufficient condition for convergence can be rewritten as

$$\|\widehat{\boldsymbol{c}}_{k+1}\|^2 < \|\widehat{\boldsymbol{c}}_k\|^2 \quad \forall k.$$

Let us then require that the absolute value of each component of the vector $\widehat{\boldsymbol{c}}_{k+1}$ is less than that of $\widehat{\boldsymbol{c}}_k$:

$$|1 - \mu\lambda_i| < 1, \quad i = 1, 2, \ldots, N.$$

The condition

$$|1 - \mu\lambda_i| < 1, \quad i = 1, 2, \ldots, N,$$

is equivalent to

$$0 < \mu < \frac{2}{\lambda_{\max}}$$

where $\lambda_{\max}$ is the *maximum eigenvalue* of R. In practice, even a stronger condition is (often) used:

$$0 < \mu < \frac{2}{\mathrm{tr}\{R\}},$$

where $\mathrm{tr}\{R\} > \lambda_{\max}$.

Normalized LMS

A promising variant of LMS is the so-called *Normalized LMS (NLMS)* algorithm:

$$\boldsymbol{w}_{k+1} = \boldsymbol{w}_k + \frac{\mu}{\|\boldsymbol{x}_k\|^2}\boldsymbol{x}_k e_k^*, \quad e_k = d_k - \boldsymbol{w}_k^H \boldsymbol{x}_k \leftarrow \text{NLMS alg.}$$

The sufficient condition for convergence:

$$0 < \mu < 2.$$

In practice, at some time points $\|\boldsymbol{x}_k\|$ can be very small. To make the NLMS algorithm more robust, we can modify it as follows:

$$\boldsymbol{w}_{k+1} = \boldsymbol{w}_k + \frac{\mu}{\|\boldsymbol{x}_k\|^2 + \delta}\boldsymbol{x}_k e_k^*,$$

so that the gain constant cannot go to infinity.

Recursive Least Squares

Idea of the Recursive Least Squares (RLS) algorithm: use sample estimate $\widehat{R}_k$ (instead of true covariance matrix R) in the equation for the weight vector and find $\boldsymbol{w}_{k+1}$ as an *update* to $\boldsymbol{w}_k$. Let

$$\begin{aligned}\widehat{R}_{k+1} &= \lambda \widehat{R}_k + \boldsymbol{x}_{k+1}\boldsymbol{x}_{k+1}^H \\ \widehat{\boldsymbol{r}}_{k+1} &= \lambda \widehat{\boldsymbol{r}}_k + \boldsymbol{x}_{k+1} d_{k+1}^*,\end{aligned}$$

where $\lambda \leq 1$ is the (so-called) *forgetting factor*. Using the *matrix inversion lemma*, we obtain

$$\begin{aligned}\widehat{R}_{k+1}^{-1} &= (\lambda \widehat{R}_k + \boldsymbol{x}_{k+1}\boldsymbol{x}_{k+1}^H)^{-1} \\ &= \frac{1}{\lambda}\left[\widehat{R}_k^{-1} - \frac{\widehat{R}_k^{-1}\boldsymbol{x}_{k+1}\boldsymbol{x}_{k+1}^H \widehat{R}_k^{-1}}{\lambda + \boldsymbol{x}_{k+1}^H \widehat{R}_k^{-1}\boldsymbol{x}_{k+1}}\right].\end{aligned}$$

Therefore,

$$\begin{aligned}
\boldsymbol{w}_{k+1} &= \widehat{R}_{k+1}^{-1}\widehat{\boldsymbol{r}}_{k+1} = \left[\widehat{R}_k^{-1} - \frac{\widehat{R}_k^{-1}\boldsymbol{x}_{k+1}\boldsymbol{x}_{k+1}^H\widehat{R}_k^{-1}}{\lambda + \boldsymbol{x}_{k+1}^H\widehat{R}_k^{-1}\boldsymbol{x}_{k+1}}\right]\widehat{\boldsymbol{r}}_k \\
&\quad + \frac{1}{\lambda}\left[\widehat{R}_k^{-1} - \frac{\widehat{R}_k^{-1}\boldsymbol{x}_{k+1}\boldsymbol{x}_{k+1}^H\widehat{R}_k^{-1}}{\lambda + \boldsymbol{x}_{k+1}^H\widehat{R}_k^{-1}\boldsymbol{x}_{k+1}}\right]\boldsymbol{x}_{k+1}d_{k+1}^* \\
&= \boldsymbol{w}_k - \boldsymbol{g}_{k+1}\boldsymbol{x}_{k+1}^H\boldsymbol{w}_k + \boldsymbol{g}_{k+1}d_{k+1}^*,
\end{aligned}$$

where

$$\boldsymbol{g}_{k+1} = \frac{\widehat{R}_k^{-1}\boldsymbol{x}_{k+1}}{\lambda + \boldsymbol{x}_{k+1}^H\widehat{R}_k^{-1}\boldsymbol{x}_{k+1}}.$$

Hence, the updating equation for the weight vector is

$$\begin{aligned}
\boldsymbol{w}_{k+1} &= \boldsymbol{w}_k - \boldsymbol{g}_{k+1}\boldsymbol{x}_{k+1}^H \boldsymbol{w}_k + \boldsymbol{g}_{k+1} d_{k+1}^* \\
&= \boldsymbol{w}_k + \boldsymbol{g}_{k+1} \underbrace{(d_{k+1}^* - \boldsymbol{x}_{k+1}^H \boldsymbol{w}_k)}_{e_{k,k+1}^*} \\
&= \boldsymbol{w}_k + \boldsymbol{g}_{k+1} e_{k,k+1}^*.
\end{aligned}$$

RLS algorithm:

- Initialization: $\boldsymbol{w}_0 = \mathbf{0}, P_0 = \delta^{-1} I$
- For each $k = 1, 2, \ldots$, compute:

$$
\begin{aligned}
\boldsymbol{h}_k &= P_{k-1} \boldsymbol{x}_k, \\
\alpha_k &= 1/(\lambda + \boldsymbol{h}_k^H \boldsymbol{x}_k), \\
\boldsymbol{g}_k &= \boldsymbol{h}_k \alpha_k, \\
P_k &= \lambda^{-1} \big[P_{k-1} - \boldsymbol{g}_k \boldsymbol{h}_k^H \big], \\
e_{k-1,k} &= d_k - \boldsymbol{w}_{k-1}^H \boldsymbol{x}_k, \\
\boldsymbol{w}_k &= \boldsymbol{w}_{k-1} + \boldsymbol{g}_k e_{k-1,k}^*, \\
e_k &= d_k - \boldsymbol{w}_k^H \boldsymbol{x}_k.
\end{aligned}
$$

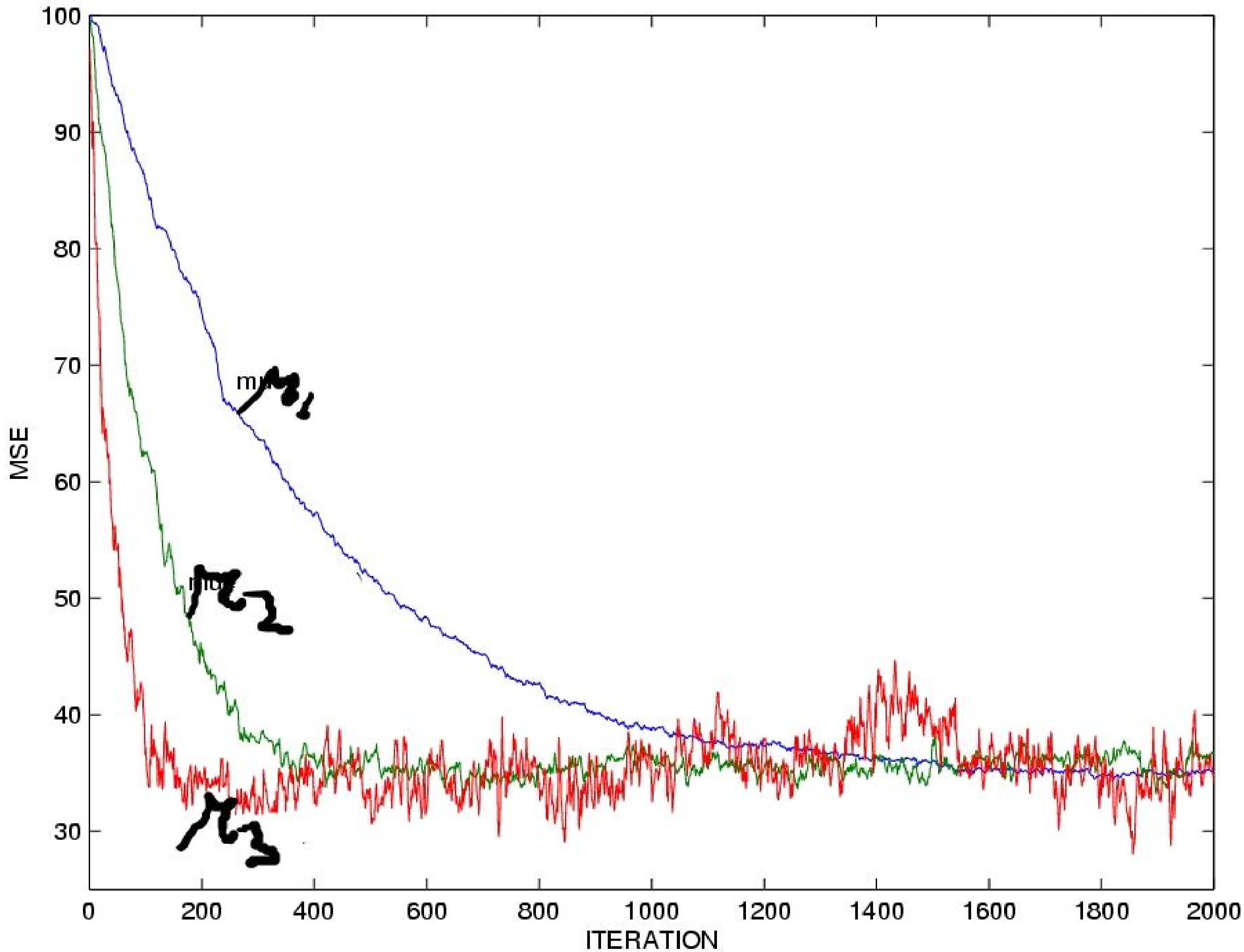
Example

LMS linear predictor of the signal

$$x(n) = 10e^{j2\pi fn} + e(n)$$

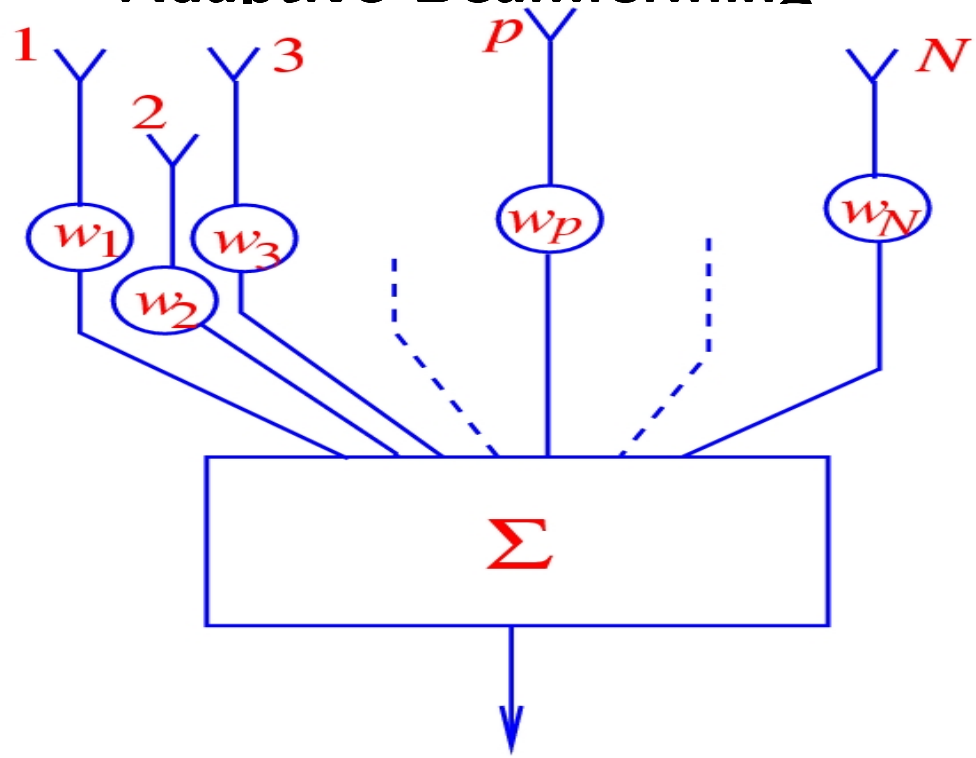
where $f = 0.1$ and

- $N = 8$,
- $e(n)$ is circular unit-variance white noise,
- $\mu_1 = 1/[10\,\mathrm{tr}(R)]$, $\mu_2 = 1/[3\,\mathrm{tr}(R)]$, $\mu_3 = 1/[\mathrm{tr}(R)]$.



100
90
80
70
60
50
40
30
MSE
0
200
400
600
800
1000
1200
1400
1600
1800
2000
ITERATION
μ_1
μ_2
μ_3

Adaptive Beamforming



The above scheme describes *narrowband beamforming,* i.e.

- *conventional beamforming* if $w_1, \ldots, w_N$ do not depend on the input/output array signals,

- *adaptive beamforming* if $w_1, \ldots, w_N$ are determined and optimized based on the input/output array signals.

Input array signal vector:

$$\boldsymbol{x}(i) = \begin{bmatrix} x_1(i) \\ x_2(i) \\ \vdots \\ x_N(i) \end{bmatrix}.$$

Complex beamformer output:

$$y(i) = \boldsymbol{w}^H \boldsymbol{x}(i).$$

Adaptive Beamforming (cont.)

Input array signal vector:

$$\boldsymbol{x}(k) = \begin{bmatrix} x_1(k) \\ x_2(k) \\ \vdots \\ x_N(k) \end{bmatrix}.$$

Complex beamformer output:

$$\begin{aligned} y(k) &= \boldsymbol{w}^H \boldsymbol{x}(k), \\ \boldsymbol{x}(k) &= \underbrace{\boldsymbol{x}_{\mathrm{s}}(k)}_{\text{signal}} + \underbrace{\boldsymbol{x}_{\mathrm{N}}(k)}_{\text{noise}} + \underbrace{\boldsymbol{x}_{\mathrm{I}}(k)}_{\text{interference}}. \end{aligned}$$

The *goal* is to filter out $\boldsymbol{x}_{\mathrm{I}}$ and $\boldsymbol{x}_{\mathrm{N}}$ as much as possible and, therefore, to obtain an approximation $\widehat{\boldsymbol{x}}_{\mathrm{S}}$ of $\boldsymbol{x}_{\mathrm{S}}$. Most popular criteria of adaptive beamforming:

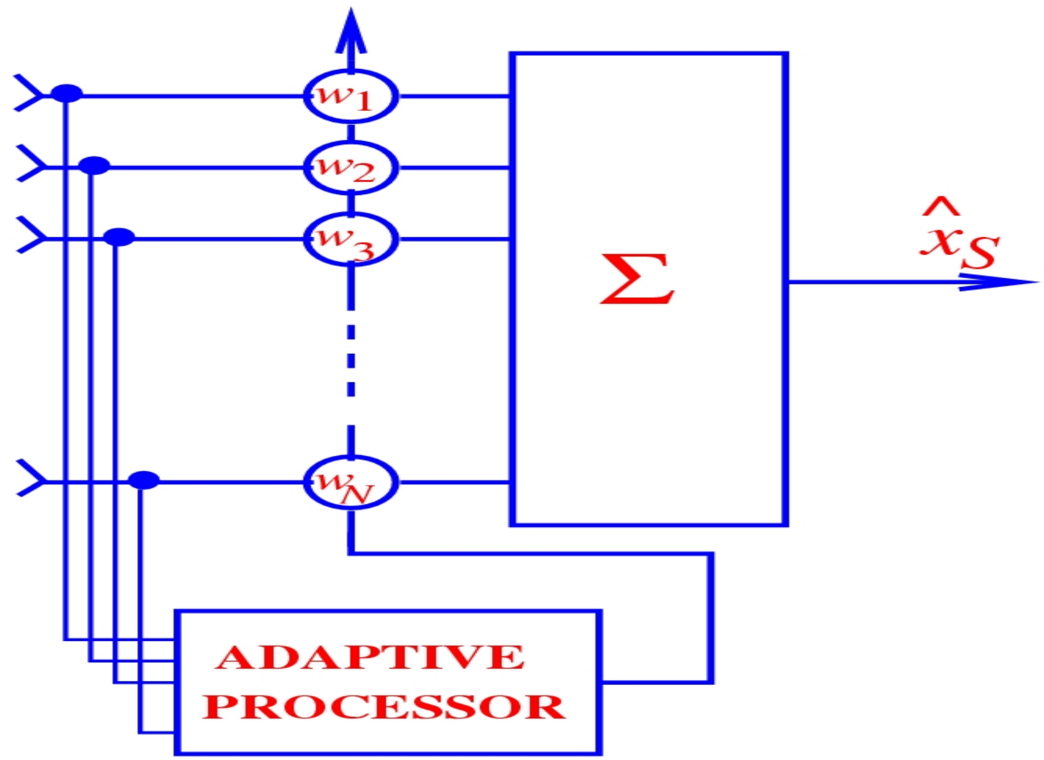
- MSE minimum

$$\min_{\boldsymbol{w}} \mathsf{MSE}, \quad \mathrm{MSE} = \mathrm{E}\left\{|d(i) - \boldsymbol{w}^H \boldsymbol{x}(i)|^2\right\}.$$

- Signal-to-Interference-plus-Noise-Ratio (SINR)

$$\max_{\boldsymbol{w}} \mathsf{SINR}, \quad \mathrm{SINR} = \frac{\mathrm{E}\left\{|\boldsymbol{w}^H \boldsymbol{x}_{\mathrm{s}}|^2\right\}}{\mathrm{E}\left\{|\boldsymbol{w}^H (\boldsymbol{x}_{\mathrm{I}} + \boldsymbol{x}_{\mathrm{N}})|^2\right\}}.$$

Adaptive Beamforming (cont.)



Adaptive Beamforming (cont.)

In the sequel, we consider the *max SINR* criterion. Rewrite the snapshot model as

$$\boldsymbol{x}(k) = s(k)\boldsymbol{a}_{\mathrm{s}} + \boldsymbol{x}_{\mathrm{I}}(k) + \boldsymbol{x}_{\mathrm{N}}(k),$$

where $\boldsymbol{a}_{\mathrm{S}}$ is the *known steering vector* of the desired signal. Then

$$\mathrm{SINR} = \frac{\sigma_{\mathrm{s}}^2 |\boldsymbol{w}^H \boldsymbol{a}_{\mathrm{s}}|^2}{\boldsymbol{w}^H \mathrm{E}\left\{(\boldsymbol{x}_{\mathrm{I}} + \boldsymbol{x}_{\mathrm{N}})(\boldsymbol{x}_{\mathrm{I}} + \boldsymbol{x}_{\mathrm{N}})^H\right\}\boldsymbol{w}} = \frac{\sigma_{\mathrm{s}}^2 |\boldsymbol{w}^H \boldsymbol{a}_{\mathrm{s}}|^2}{\boldsymbol{w}^H R \boldsymbol{w}}$$

where

$$R = \mathrm{E}\left\{(\boldsymbol{x}_{\mathrm{I}} + \boldsymbol{x}_{\mathrm{N}})(\boldsymbol{x}_{\mathrm{I}} + \boldsymbol{x}_{\mathrm{N}})^H\right\}$$

is the *interference-plus-noise covariance matrix*.

Obviously, SINR does not depend on rescaling of $\boldsymbol{w}$, i.e. if $\boldsymbol{w}_{\mathrm{opt}}$ is an optimal weight, then $\alpha \boldsymbol{w}_{\mathrm{opt}}$ is such a vector too. Therefore, max SINR is equivalent to

$$\min_{\boldsymbol{w}} \boldsymbol{w}^H R \boldsymbol{w} \quad \text{subject to} \quad \boldsymbol{w}^H \boldsymbol{a}_{\mathrm{S}} = \mathrm{const.}$$

Let $\mathrm{const} = 1$. Then

$$\begin{aligned} H(\boldsymbol{w}) &= \boldsymbol{w}^H R \boldsymbol{w} + \lambda(1 - \boldsymbol{w}^H \boldsymbol{a}_{\mathrm{s}}) + \lambda^*(1 - \boldsymbol{a}_{\mathrm{s}}^H \boldsymbol{w}) \\ \nabla_{\boldsymbol{w}} H(\boldsymbol{w}) &= (R\boldsymbol{w} - \lambda \boldsymbol{a}_{\mathrm{s}})^* = \mathbf{0} \quad \Longrightarrow \\ R\boldsymbol{w} &= \lambda \boldsymbol{a}_{\mathrm{s}} \Longrightarrow \boldsymbol{w}_{\mathrm{opt}} = \lambda R^{-1} \boldsymbol{a}_{\mathrm{s}}. \end{aligned}$$

This is a *spatial version* of the Wiener-Hopf equation!

From the constraint equation, we obtain

$$\lambda = \frac{1}{\boldsymbol{a}_{\mathrm{s}}^H R^{-1} \boldsymbol{a}_{\mathrm{s}}}$$

and therefore

$$\boldsymbol{w}_{\mathrm{opt}} = \frac{1}{\boldsymbol{a}_{\mathrm{s}}^H R^{-1} \boldsymbol{a}_{\mathrm{s}}} R^{-1} \boldsymbol{a}_{\mathrm{s}} \quad \longleftarrow \text{MVDR beamformer.}$$

Substituting $\boldsymbol{w}_{\mathrm{opt}}$ into the SINR expression, we obtain

$$\max \mathrm{SINR} = \mathrm{SINR}_{\mathrm{opt}} = \frac{\sigma_{\mathrm{s}}^2 (\boldsymbol{a}_{\mathrm{s}}^H R^{-1} \boldsymbol{a}_{\mathrm{s}})^2}{\boldsymbol{a}_{\mathrm{s}}^H R^{-1} R R^{-1} \boldsymbol{a}_{\mathrm{s}}} = \sigma_{\mathrm{s}}^2 \boldsymbol{a}_{\mathrm{s}}^H R^{-1} \boldsymbol{a}_{\mathrm{s}}.$$

If there are no interference sources (only white noise with variance σ^2):

$$\mathrm{SINR}_{\mathrm{opt}} = \frac{\sigma_{\mathrm{s}}^2}{\sigma^2} \boldsymbol{a}_{\mathrm{s}}^H \boldsymbol{a}_{\mathrm{s}} = \frac{N \sigma_{\mathrm{s}}^2}{\sigma^2}.$$

Adaptive Beamforming (cont.)

Let us study what happens with the optimal SINR if the covariance matrix includes the signal component:

$$R_x = \mathrm{E}\left\{\boldsymbol{x}\boldsymbol{x}^H\right\} = R + \sigma_{\mathrm{s}}^2 \boldsymbol{a}_{\mathrm{s}} \boldsymbol{a}_{\mathrm{s}}^H.$$

Using the matrix inversion lemma, we have

$$\begin{aligned}
R_x^{-1}\boldsymbol{a}_{\mathrm{s}} &= (R + \sigma_{\mathrm{s}}^2 \boldsymbol{a}_{\mathrm{s}} \boldsymbol{a}_{\mathrm{S}}^H)^{-1}\boldsymbol{a}_{\mathrm{s}} \\
&= \left(R^{-1} - \frac{R^{-1}\boldsymbol{a}_{\mathrm{s}}\boldsymbol{a}_{\mathrm{s}}^H R^{-1}}{1/\sigma_{\mathrm{s}}^2 + \boldsymbol{a}_{\mathrm{s}}^H R^{-1}\boldsymbol{a}_{\mathrm{s}}}\right)\boldsymbol{a}_{\mathrm{s}} \\
&= \left(1 - \frac{\boldsymbol{a}_{\mathrm{s}}^H R^{-1}\boldsymbol{a}_{\mathrm{s}}}{1/\sigma_{\mathrm{s}}^2 + \boldsymbol{a}_{\mathrm{s}}^H R^{-1}\boldsymbol{a}_{\mathrm{s}}}\right) R^{-1}\boldsymbol{a}_{\mathrm{s}} \\
&= \alpha R^{-1}\boldsymbol{a}_{\mathrm{s}}.
\end{aligned}$$

Optimal SINR is not affected!

However, the above result holds only if

- there is an *infinite* number of snapshots and
- $\boldsymbol{a}_{\mathrm{S}}$ is known *exactly*.

Adaptive Beamforming (cont.)

Gradient algorithm maximizing SNR (very similar to LMS):

$$\boldsymbol{w}_{k+1} = \boldsymbol{w}_k + \mu(\boldsymbol{a}_{\rm s} - \boldsymbol{x}_k \boldsymbol{x}_k^H \boldsymbol{w}_k),$$

where, again, we use the simple notation $\boldsymbol{w}_k = \boldsymbol{w}(k)$ and $\boldsymbol{x}_k = \boldsymbol{x}(k)$. The vector $\boldsymbol{w}_k$ converges to $\boldsymbol{w}_{\rm opt} \sim R^{-1}\boldsymbol{a}_{\rm s}$ if

$$0 < \mu < \frac{2}{\lambda_{\max}} \quad \Longrightarrow \quad 0 < \mu < \frac{2}{\mathrm{tr}\{R\}}.$$

The disadvantage of the gradient algorithms is that the convergence may be very slow, i.e. it depends on the *eigenvalue spread* of R.

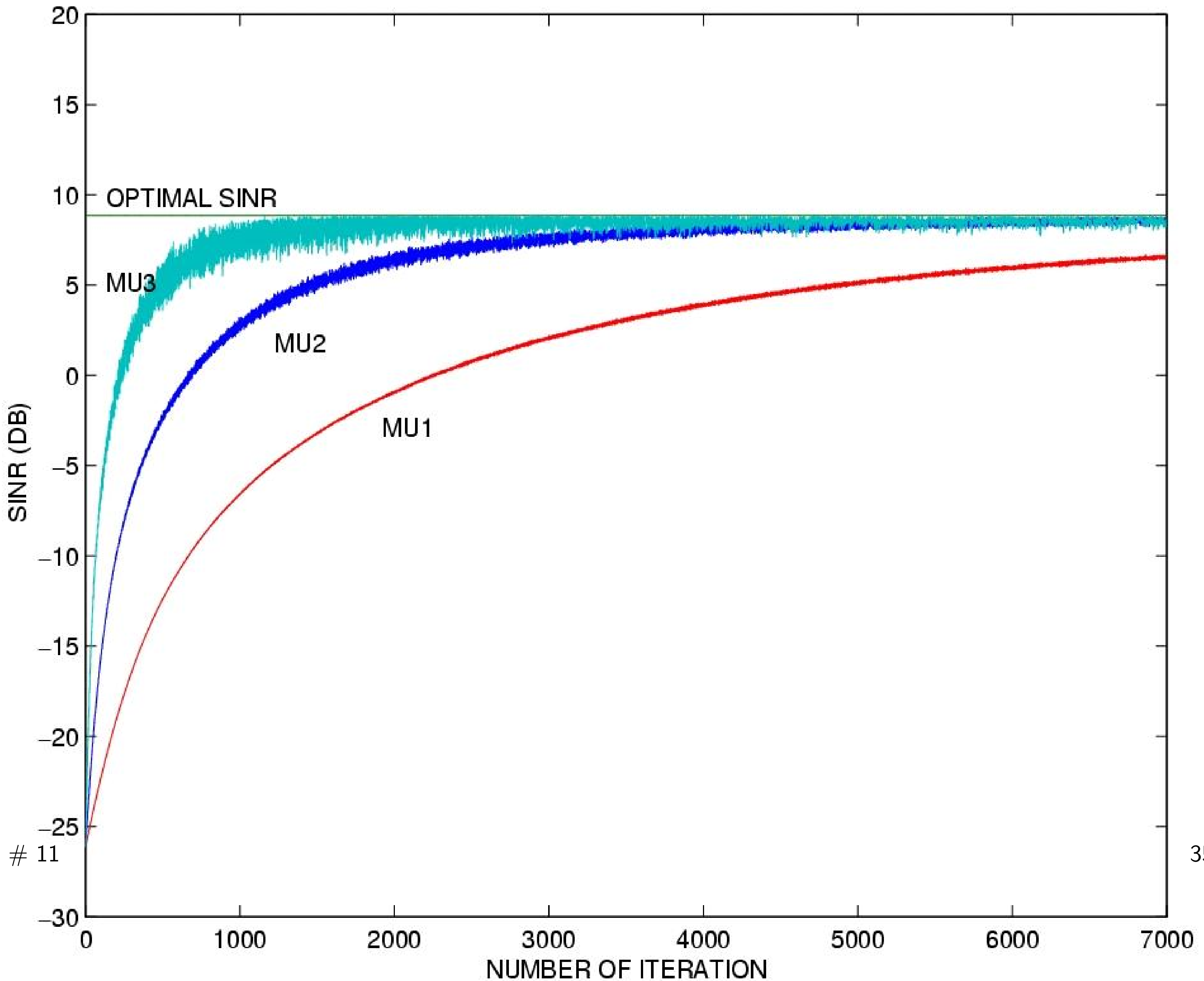
Example

- $N = 8$,

- single signal from $\theta_{\mathrm{s}} = 0^\circ$ and $\mathrm{SNR} = 0$ dB,

- single interference from $\theta_{\mathrm{I}} = 30^\circ$ and INR$= 40$ dB,

- $\mu_1 = 1/[50\,\mathrm{tr}(R)]$, $\mu_2 = 1/[15\,\mathrm{tr}(R)]$, $\mu_3 = 1/[5\,\mathrm{tr}(R)]$.


OPTIMAL SINR
MU3
MU2
MU1
SINR (DB)
NUMBER OF ITERATION
20
15
10
5
0
−5
−10
−15
−20
−25
−30
0
1000
2000
3000
4000
5000
6000
7000

Adaptive Beamforming (cont.)

Sample Matrix Inversion (SMI) Algorithm:

$$\boldsymbol{w}_{\mathrm{SMI}} = \widehat{R}^{-1}\boldsymbol{a}_{\mathrm{S}},$$

where $\widehat{R}$ is the sample covariance matrix

$$\widehat{R} = \frac{1}{K}\sum_{k=1}^{K} \boldsymbol{x}_k \boldsymbol{x}_k^H.$$

Reed-Mallet-Brennan (RMB) rule: under mild conditions, the mean losses (relative to the optimal SINR) due to the SMI approximation of $\boldsymbol{w}_{\mathrm{opt}}$ do not exceed 3 dB if

$$K \geq 2N.$$

Hence, the SMI provides very fast convergence rate, in general.

Adaptive Beamforming (cont.)

Loaded SMI:

$$\boldsymbol{w}_{\mathrm{LSMI}} = \widehat{R}_{\mathrm{DL}}^{-1}\boldsymbol{a}_{\mathrm{S}}, \quad \widehat{R}_{\mathrm{DL}} = \widehat{R} + \gamma I,$$

where the optimal weight $\gamma \approx 2\sigma^2$. LSMI allows convergence faster than N snapshots!

LSMI convergence rule: under mild conditions, the mean losses (relative to the optimal SINR) due to the LSMI approximation of $\boldsymbol{w}_{\mathrm{opt}}$ do not exceed few dB's if

$$K \geq L$$

where L is the number of interfering sources. Hence, the LSMI provides faster convergence rate than SMI (usually, $2N \gg L$)!

Example

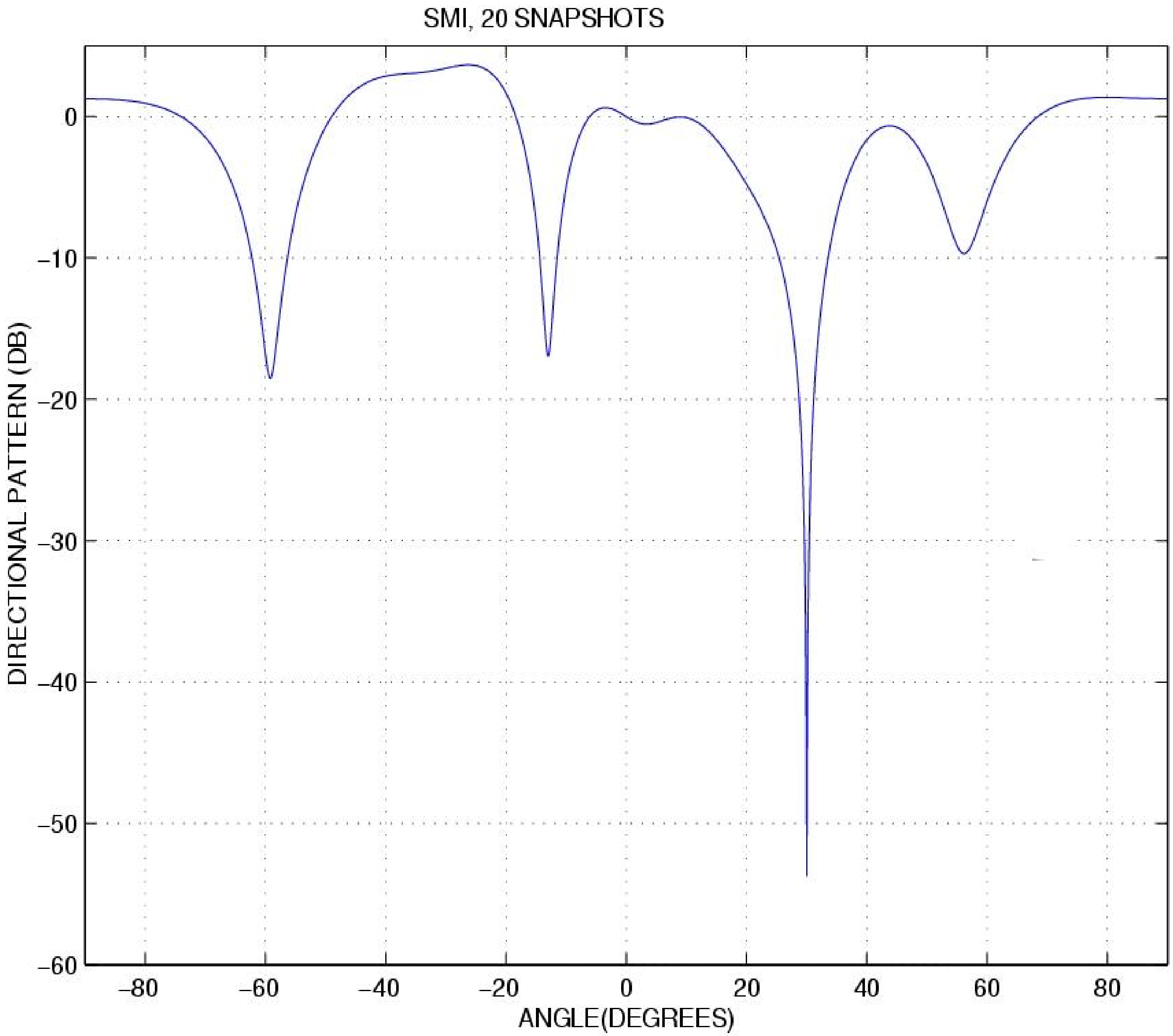
- $N = 10$,

- single signal from $\theta_{\mathrm{s}} = 0^\circ$ and $\mathrm{SNR} = 0$ dB,

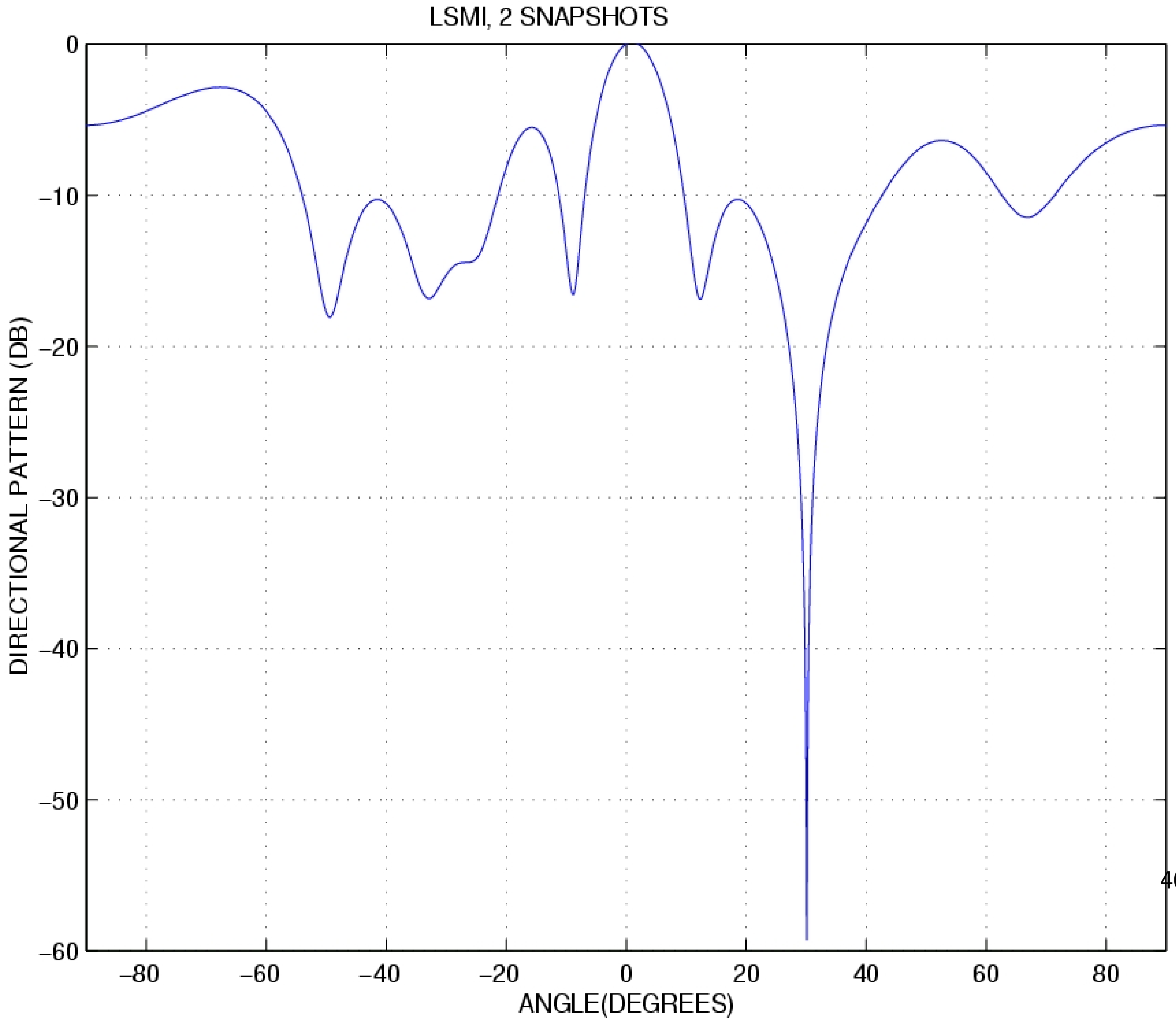
- single interference from $\theta_{\mathrm{I}} = 30^\circ$ and INR$= 40$ dB,

- SMI vs. LSMI.

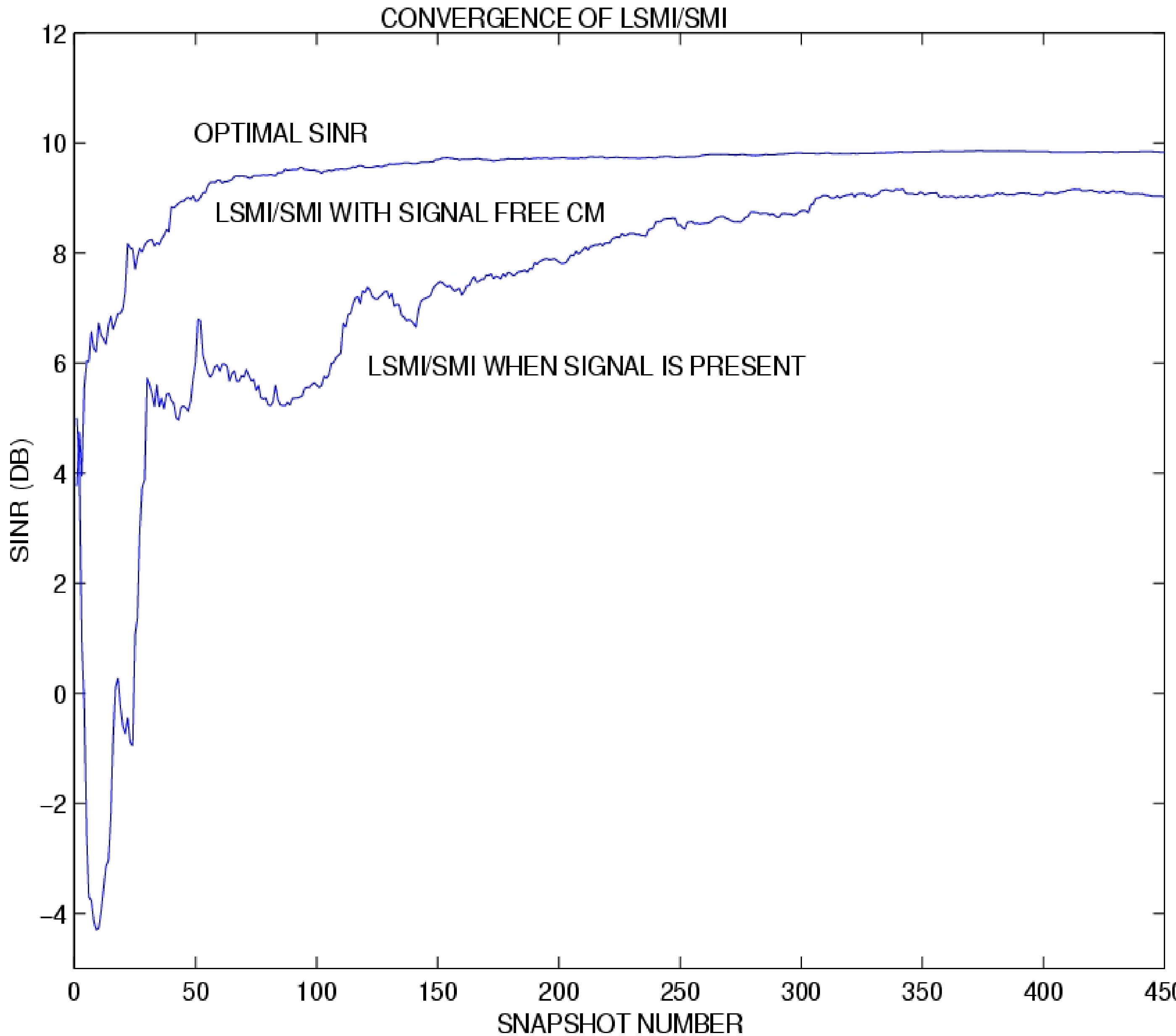
SMI directional pattern (signal free case), $K = 20$



LSMI directional pattern (signal free case), $K = 2$



Convergence rates with signal absent and present:



Adaptive Beamforming (cont.)

Hung-Turner (Projection) Algorithm:

$$\boldsymbol{w}_{\mathrm{HT}} = (I - X(X^H X)^{-1} X^H)\boldsymbol{a}_{\mathrm{S}},$$

i.e. data-orthogonal projection is used instead of inverse covariance matrix. For Hung-Turner method, a satisfactory performance is achieved with

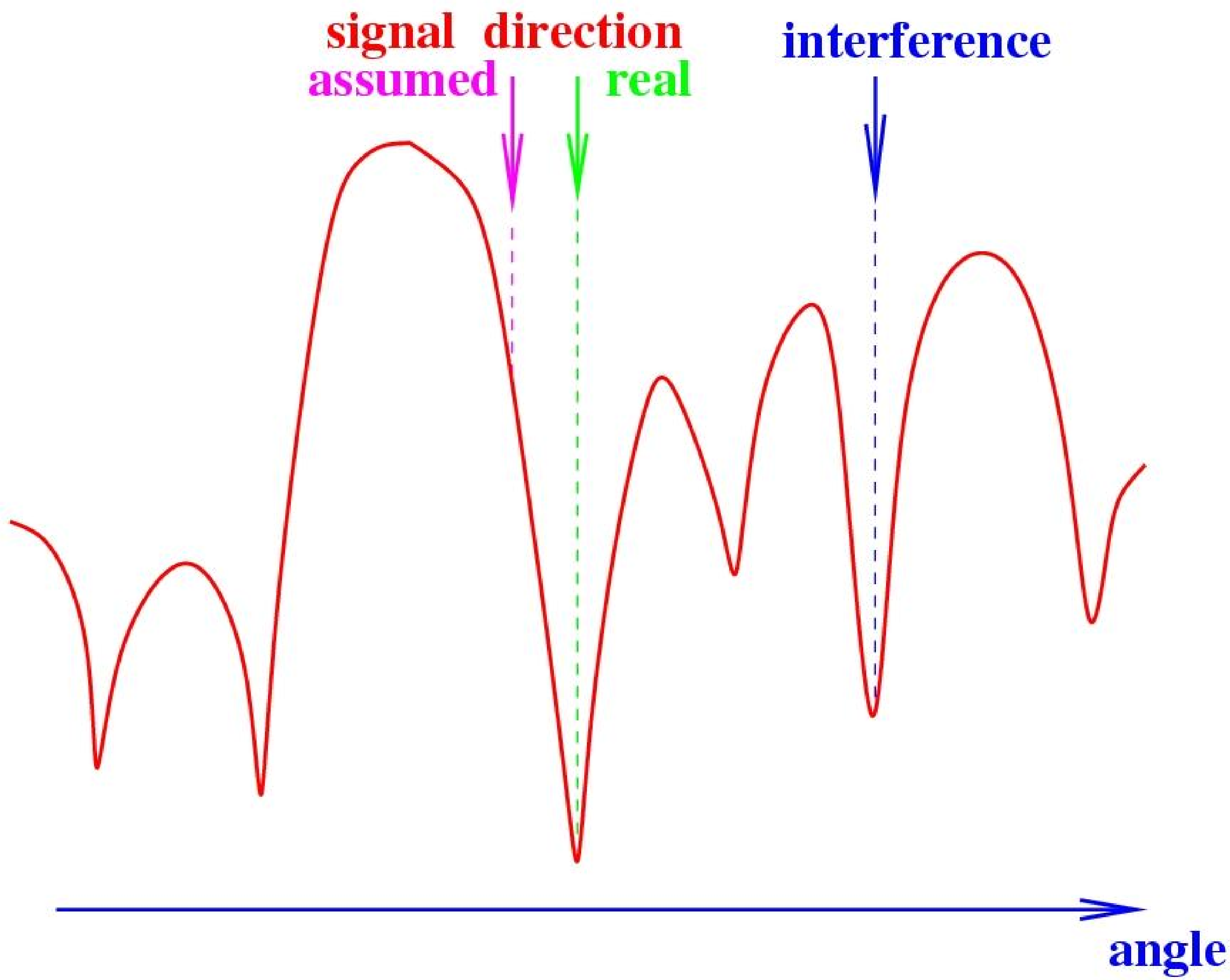
$$K \geq L.$$

Optimal value of K

$$K_{\mathrm{opt}} = \sqrt{(N+1)L} - 1.$$

Drawback: number of sources should be known *a priori*.

Look direction mismatch (pointing error) problem:



This effect is sometimes referred to as the *signal cancellation phenomenon*. Additional constraints are required to stabilize the mean beam response

$$\min_{\boldsymbol{w}} \boldsymbol{w}^H R \boldsymbol{w} \quad \text{subject to} \quad C^H \boldsymbol{w} = \boldsymbol{f}.$$

1. Point constraints: Matrix of constrained directions:

$$C = [\boldsymbol{a}_{\mathrm{S},1}, \boldsymbol{a}_{\mathrm{S},2} \cdots \boldsymbol{a}_{\mathrm{S},M}],$$

where $\boldsymbol{a}_{\mathrm{S},i}$ are all taken in the neighborhood of $\boldsymbol{a}_{\mathrm{S}}$ and include $\boldsymbol{a}_{\mathrm{S}}$ as well. Vector of constraints:

$$\boldsymbol{f} = \begin{bmatrix} 1 \\ 1 \\ \vdots \\ 1 \end{bmatrix}.$$

2. Derivative constraints: Matrix of constrained directions:

$$C = \left[\boldsymbol{a}_{\mathrm{S}}, \left.\frac{\partial \boldsymbol{a}(\theta)}{\partial \theta}\right|_{\theta=\theta_{\mathrm{S}}}, \cdots, \left.\frac{\partial^{M-1} \boldsymbol{a}(\theta)}{\partial \theta^{M-1}}\right|_{\theta=\theta_{\mathrm{S}}} \right],$$

where $\boldsymbol{a}_{\mathrm{S},i}$ are all taken in the neighborhood of $\boldsymbol{a}_{\mathrm{S}}$ and include $\boldsymbol{a}_{\mathrm{S}}$ as well. Vector of constraints:

$$\boldsymbol{f} = \begin{bmatrix} 1 \\ 0 \\ \vdots \\ 0 \end{bmatrix}.$$

Note that

$$\left.\frac{\partial^{k} \boldsymbol{a}(\theta)}{\partial \theta^{k}}\right|_{\theta=\theta_{\mathrm{S}}} = D^{k} \boldsymbol{a}_{\mathrm{S}},$$

where D is the matrix depending on θ_{s} and on array geometry.

Adaptive Beamforming (cont.)

$$\boldsymbol{w}_{\text{opt}} = R^{-1}C(C^H R^{-1} C)^{-1} \boldsymbol{f}$$

and its SMI version:

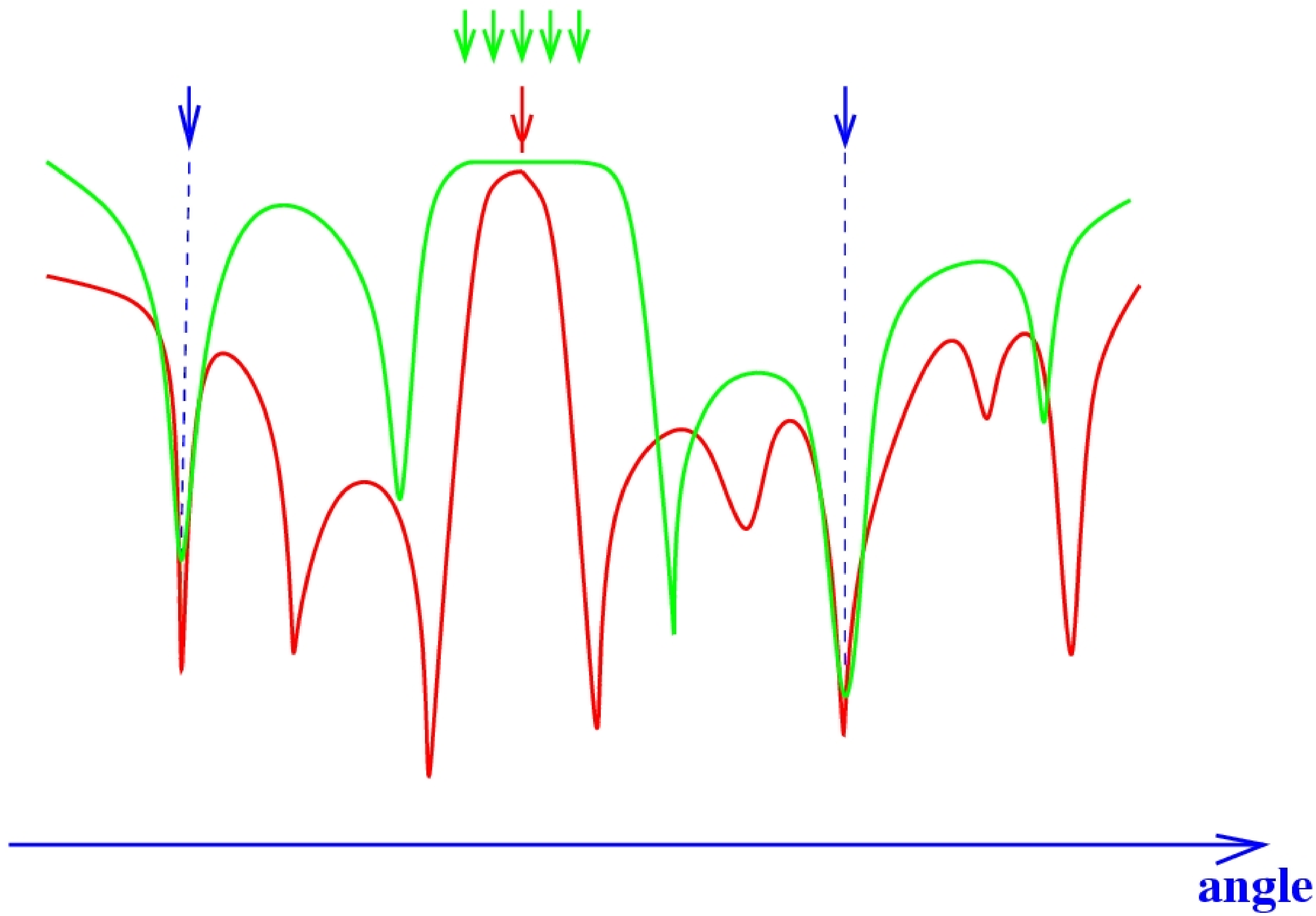
$$\boldsymbol{w}_{\text{opt}} = \widehat{R}^{-1}C(C^H \widehat{R}^{-1} C)^{-1} \boldsymbol{f}.$$

- Additional constraints "protect" the directions in the neighborhood of the assumed signal direction.

- Additional constraints require enough degrees of freedom (DOF's) – number of sensors must be large enough.

- Gradient algorithms exist for the constraint adaptation.

Effect of point constraints:



Adaptive Beamforming (cont.)

Generalized Sidelobe Canceller (GSC): Let us decompose

$$\boldsymbol{w}_{\rm opt} = R^{-1}C(C^H R^{-1}C)^{-1}\boldsymbol{f}$$

into two components, one in the constrained subspace, and one orthogonal to it:

$$\begin{aligned}
\boldsymbol{w}_{\rm opt} &= \underbrace{(P_C + P_C^{\perp})}_{I}\boldsymbol{w}_{\rm opt} \\
&= C(C^H C)^{-1}\underbrace{C^H R^{-1}C(C^H R^{-1}C)^{-1}}_{I}\boldsymbol{f} \\
&\quad + P_C^{\perp} R^{-1}C(C^H R^{-1}C)^{-1}\boldsymbol{f}.
\end{aligned}$$

Generalizing this approach, we obtain the following decomposition for $\boldsymbol{w}_{\rm opt}$:

$$\boldsymbol{w}_{\rm opt} = \boldsymbol{w}_{\rm q} - B\boldsymbol{w}_{\rm a},$$

where

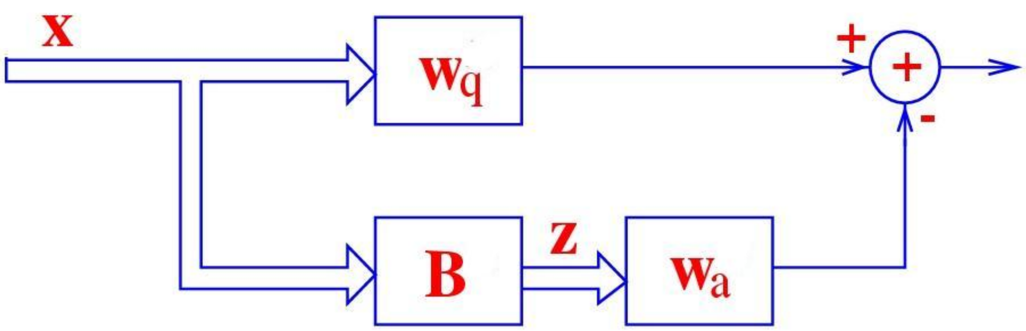
$$\boldsymbol{w}_{\rm q} = C(C^H C)^{-1}\boldsymbol{f}$$

is the so-called *quiescent* weight vector,

$$B^H C = 0,$$

B is the blocking matrix, and $\boldsymbol{w}_{\rm a}$ is the new adaptive weight vector.

Generalized Sidelobe Canceller (GSC):



- Choice of B is *not unique*. We can take $B = P_C^\perp$. However, in this case B is not of full rank. More common choice is to assume $N \times (N - M)$ full-rank matrix B. Then, the vectors $z = B^H \boldsymbol{x}$ and $\boldsymbol{w}_\mathrm{a}$ both have shorter length $(N - M) \times 1$ relative to the $N \times 1$ vectors $\boldsymbol{x}$ and $\boldsymbol{w}_\mathrm{q}$.

- Since the constrained directions are *blocked* by the matrix B, the signal cannot be suppressed and, therefore, the weight vector $\boldsymbol{w}_\mathrm{a}$ can adapt *freely* to suppress interference by minimizing the output GSC power

$$\begin{aligned} Q_{\mathrm{GSC}} &= (\boldsymbol{w}_{\mathrm{q}} - B\boldsymbol{w}_{\mathrm{a}})^H R(\boldsymbol{w}_{\mathrm{q}} - B\boldsymbol{w}_{\mathrm{a}}) \\ &= \boldsymbol{w}_{\mathrm{q}}^H R\boldsymbol{w}_{\mathrm{q}} - \boldsymbol{w}_{\mathrm{q}}^H RB\boldsymbol{w}_{\mathrm{a}} - \boldsymbol{w}_{\mathrm{a}}^H B^H R\boldsymbol{w}_{\mathrm{q}} \\ &\quad + \boldsymbol{w}_{\mathrm{a}}^H B^H RB\boldsymbol{w}_{\mathrm{a}}. \end{aligned}$$

The solution is $\boldsymbol{w}_{\mathrm{a,opt}} = (B^H RB)^{-1} B^H R\boldsymbol{w}_{\mathrm{q}}$.

Adaptive Beamforming (cont.)

Generalized Sidelobe Canceller (GSC): Noting that

$$y(k) = \boldsymbol{w}_{\mathrm{q}}^H \boldsymbol{x}(k), \quad \boldsymbol{z}(k) = B^H \boldsymbol{x}(k),$$

we obtain

$$\begin{aligned}
R_z &= \mathrm{E}\{\boldsymbol{z}(k)\boldsymbol{z}(k)^H\} \\
&= B^H \mathrm{E}\{\boldsymbol{x}(k)\boldsymbol{x}(k)^H\} B \\
&= B^H R B, \\
\boldsymbol{r}_{yz} &= \mathrm{E}\{\boldsymbol{z}(k) y^*(k)\} \\
&= B^H \mathrm{E}\{\boldsymbol{x}(k)\boldsymbol{x}(k)^H\} \boldsymbol{w}_{\mathrm{q}} \\
&= B^H R \boldsymbol{w}_{\mathrm{q}}.
\end{aligned}$$

Hence,

$$\boldsymbol{w}_{\mathrm{a,opt}} = R_z^{-1}\boldsymbol{r}_{yz} \quad \longleftarrow \text{Wiener-Hopf equation!}$$

How to Choose B?

Choose $N - M$ linearly independent vectors $\boldsymbol{b}_i$:

$$B = [\boldsymbol{b}_1 \boldsymbol{b}_2 \cdots \boldsymbol{b}_{N-M}]$$

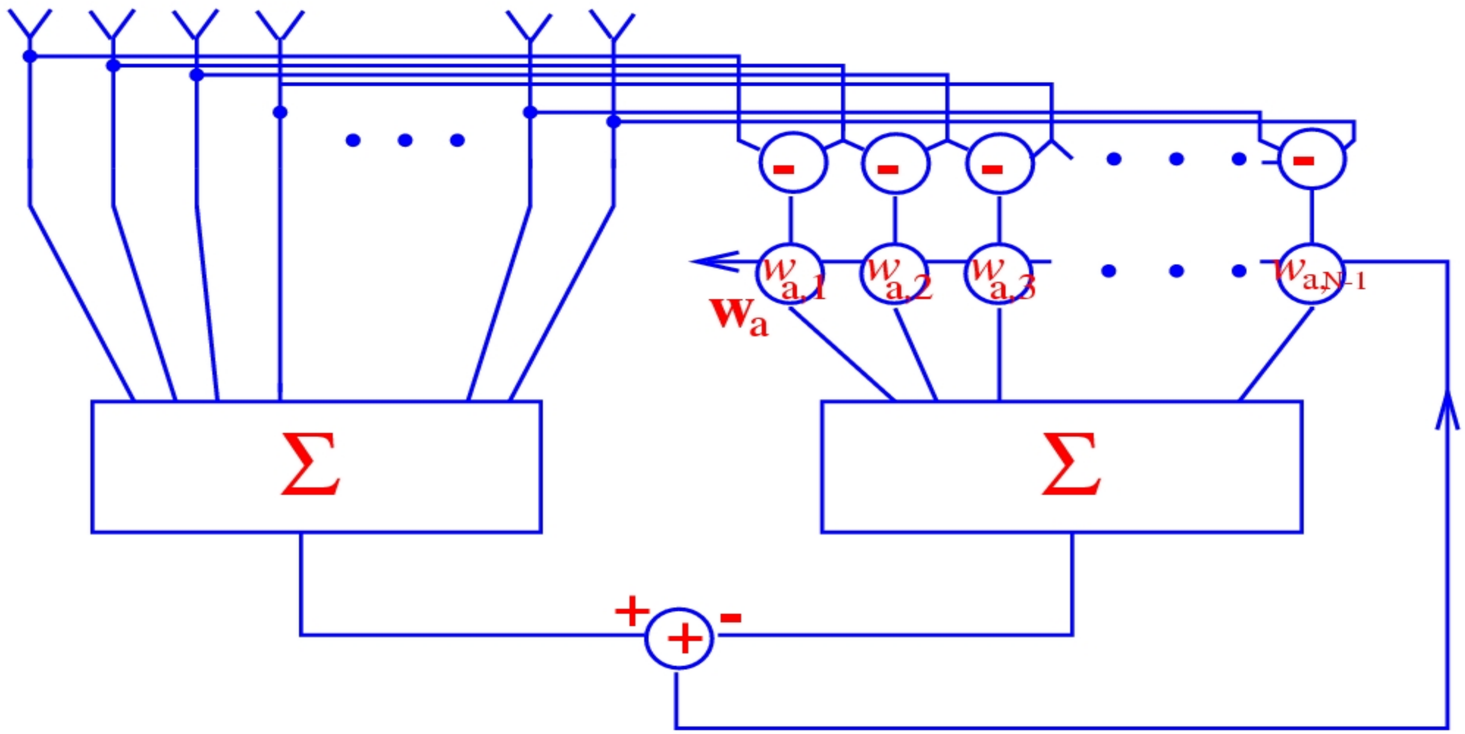
so that

$$\boldsymbol{b}_i \perp \boldsymbol{c}_k, \quad i = 1, 2, \ldots, N - M, \quad k = 1, 2, \ldots, M,$$

where $\boldsymbol{c}_k$ is the kth column of C.

There are *many* possible choices of B!

Example: GSC in the Particular Case of Normal Direction (Single) Constraint and for a Particular Choice of Blocking Matrix:



In this particular example

$$C = \begin{bmatrix} 1 \\ 1 \\ \vdots \\ 1 \end{bmatrix}.$$

$$B^H = \begin{bmatrix} 1 & -1 & 0 & \cdots & 0 & 0 \\ 0 & 1 & -1 & \cdots & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \\ 0 & 0 & 0 & \cdots & 1 & -1 \end{bmatrix},$$

and

$$\boldsymbol{x}(k) = \begin{bmatrix} x_1(k) \\ x_2(k) \\ \vdots \\ x_N(k) \end{bmatrix}, \quad \boldsymbol{z}(k) = \begin{bmatrix} x_1(k) - x_2(k) \\ x_2(k) - x_3(k) \\ \vdots \\ x_{N-1}(k) - x_N(k) \end{bmatrix}.$$

Partially Adaptive Beamforming

In many applications, number of interfering sources is much less than the number of adaptive weights [adaptive degrees of freedom (DOF's)]. In such cases, *partially adaptive arrays* can be used.

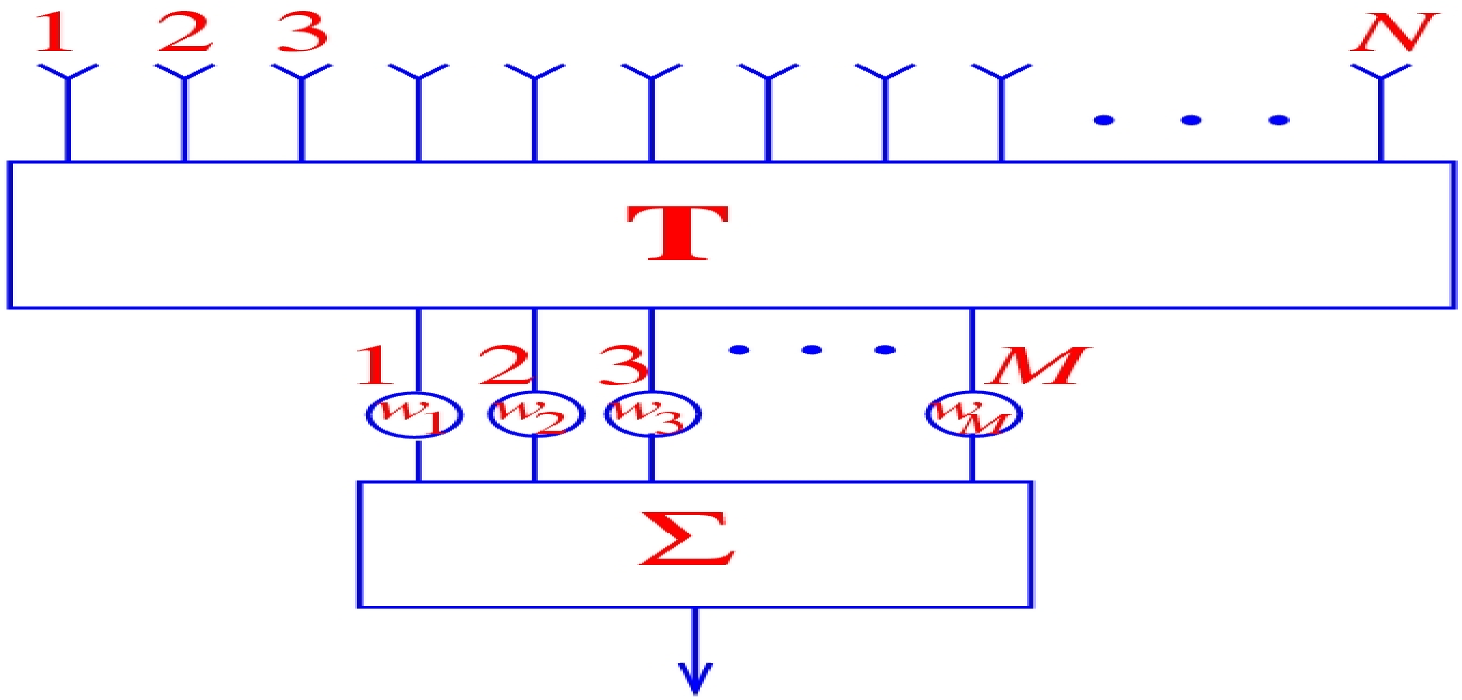
Idea: use nonadaptive preprocessor reducing the number of adaptive channels:

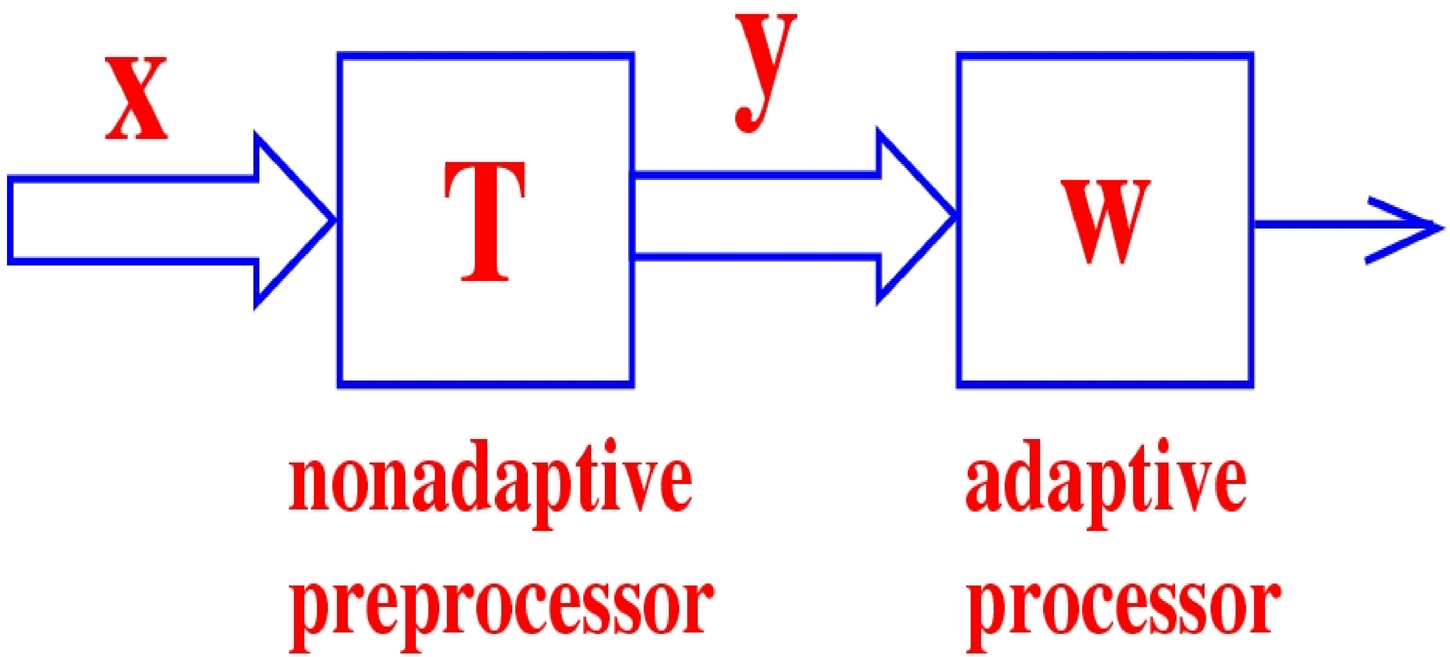
$$\boldsymbol{y}(i) = T^H \boldsymbol{x}(i),$$

where

- $\boldsymbol{y}$ has a reduced dimension $M \times 1$ $(M < N)$ compared with $N \times 1$ vector $\boldsymbol{x}$,
- T is an $N \times M$ full-rank matrix.

Partially adaptive beamformer




x
T
y
w
nonadaptive
preprocessor
adaptive
processor

Partially Adaptive Beamforming

There are two types of nonadaptive preprocessors:

- *subarray* preprocessor,
- *beamspace* preprocessor.

For arbitrary preprocessor:

$$R_y = \mathrm{E}\{\boldsymbol{y}(i)\boldsymbol{y}(i)^H\} = T^H \mathrm{E}\{\boldsymbol{x}(i)\boldsymbol{x}(i)^H\} T = T^H R T.$$

Recall the previously-used representation:

$$R = ASA^H + \sigma^2 I.$$

After the preprocessing, we have

$$\begin{aligned} R_y &= T^H A S A^H T + \sigma^2 T^H T \\ &= \widetilde{A} S \widetilde{A}^H + Q \\ \widetilde{A} &= T^H A \\ Q &= \sigma^2 T^H T. \end{aligned}$$

- Preprocessing changes array manifold.
- Preprocessing may lead to colored noise.

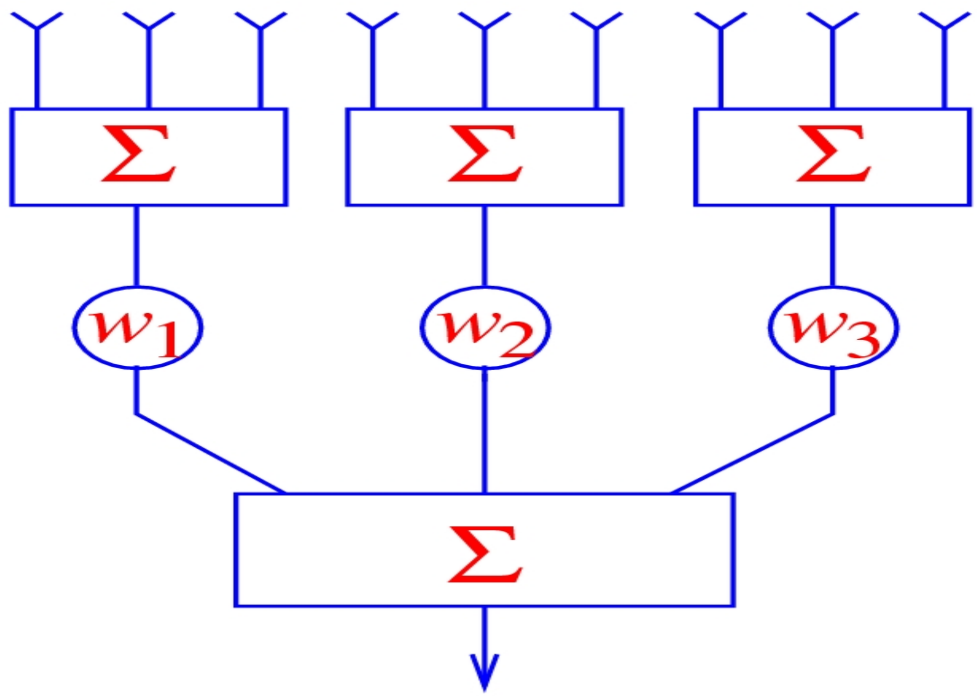
Choosing T with orthonormal columns, we have

$$T^H T = I,$$

and, therefore, the effect of colored noise may be removed.

Partially Adaptive Beamforming

Partially adaptive beamformer based on subarray preprocessing



Preprocessing matrix in this particular case:

$$T^H = \frac{1}{\sqrt{3}} \begin{bmatrix} 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 \end{bmatrix},$$

(note that $T^H T = I$ here!)

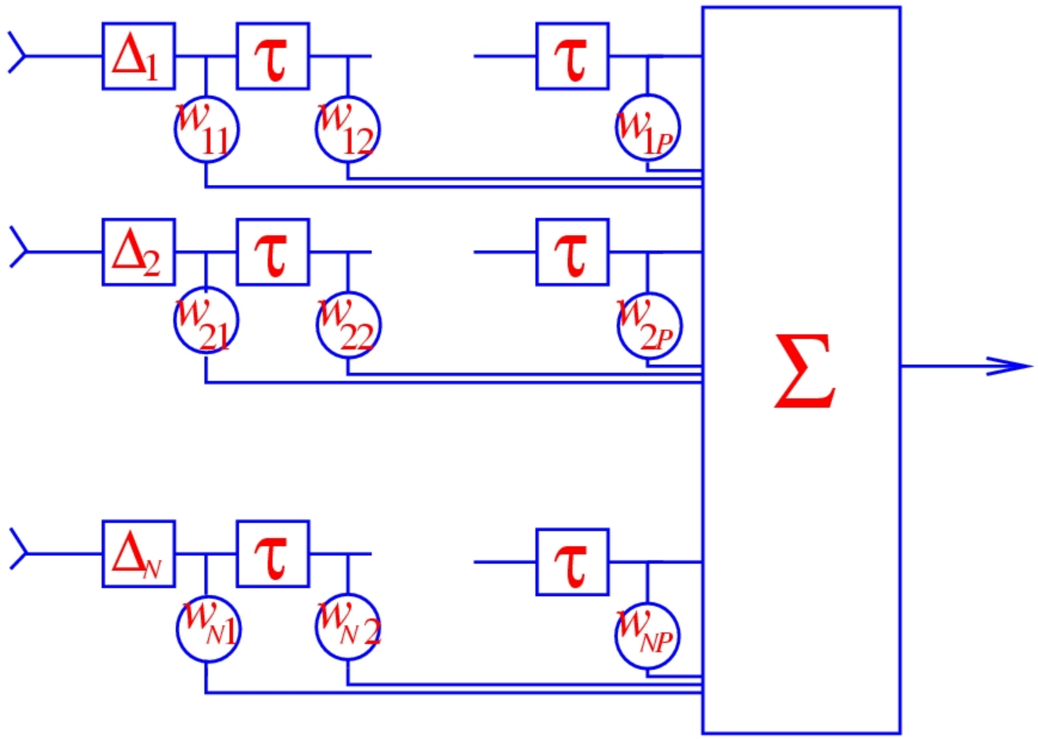
In the general case

$$T = \begin{bmatrix} \boldsymbol{a}_{\mathrm{S},1} & \mathbf{0} & \cdots & \mathbf{0} \\ \mathbf{0} & \boldsymbol{a}_{\mathrm{S},2} & \cdots & \mathbf{0} \\ \vdots & \vdots & \vdots & \vdots \\ \mathbf{0} & \cdots & \mathbf{0} & \boldsymbol{a}_{\mathrm{S},M} \end{bmatrix},$$

where $L = N/M$ is the size of each subarray, and $T^H T = I$ holds true if $\boldsymbol{a}_{\mathrm{S},k}^H \boldsymbol{a}_{\mathrm{S},k} = 1,\ k = 1, 2, \ldots, M$.

Wideband Space-Time Processing

In the *wideband* case, we must consider *joint* space-time processing:



Wideband Space-Time Processing (cont.)

Wideband case:

- Higher dimension of the problem (NP instead of N),
- Steering vector depends on frequency.

Constant Modulus Algorithm (CMA)

Application: separation of constant-modulus sources.

- Narrowband signals: the received signal is an *instantaneous linear mixture:*

$$\boldsymbol{x}_k = A\boldsymbol{s}_k.$$

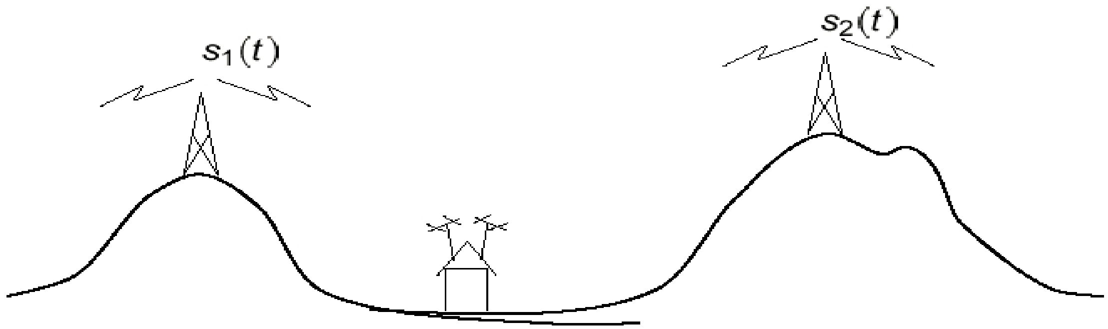
- *Objective:* find inverse W, so that

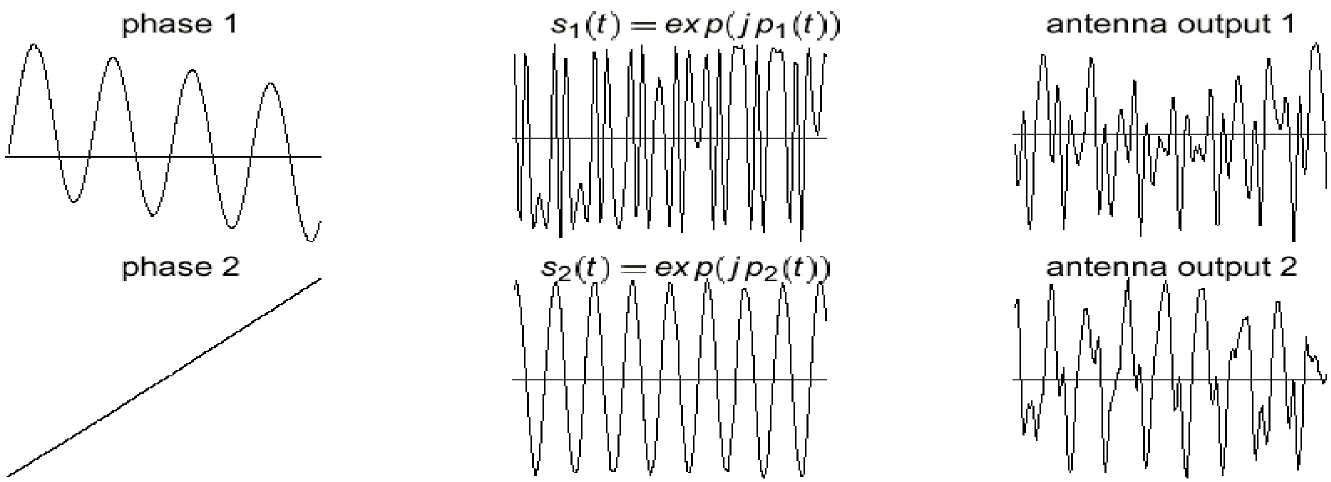
$$\boldsymbol{y}_k = W^H \boldsymbol{x}_k = \boldsymbol{s}_k.$$

  Challenge: both A and $\boldsymbol{s}_k$ are unknown!

- However, we have *side knowledge:* sources are phase modulated, i.e.

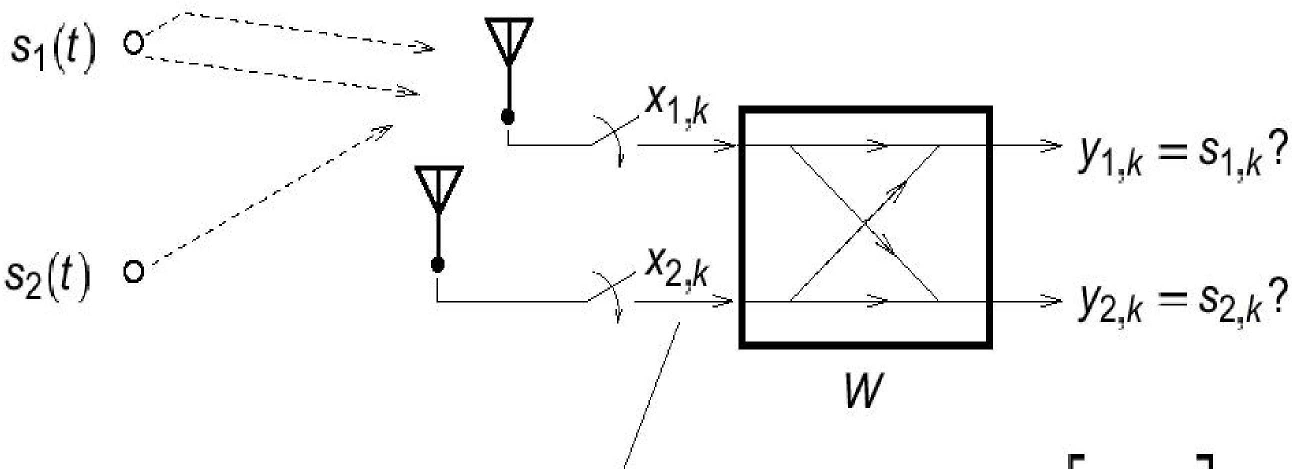
$$s_i(t) = \exp(j\phi_i(t)).$$


$s_1(t)$
$s_2(t)$


phase 1
$s_1(t) = exp(j p_1(t))$
antenna output 1
phase 2
$s_2(t) = exp(j p_2(t))$
antenna output 2

Constant Modulus Algorithm (cont.)

Simple example: 2 sources, 2 antennas.



$$
\begin{aligned}
x_{1,k} &= a_{11}s_{1,k} + a_{12}s_{2,k} \\
x_{2,k} &= a_{21}s_{1,k} + a_{22}s_{2,k}
\end{aligned}
\quad \Leftrightarrow \quad
\begin{bmatrix} x_{1,k} \\ x_{2,k} \end{bmatrix} = A \begin{bmatrix} s_{1,k} \\ s_{2,k} \end{bmatrix}
$$

Let

$$\boldsymbol{w} = \begin{bmatrix} w_1 \\ w_2 \end{bmatrix}$$

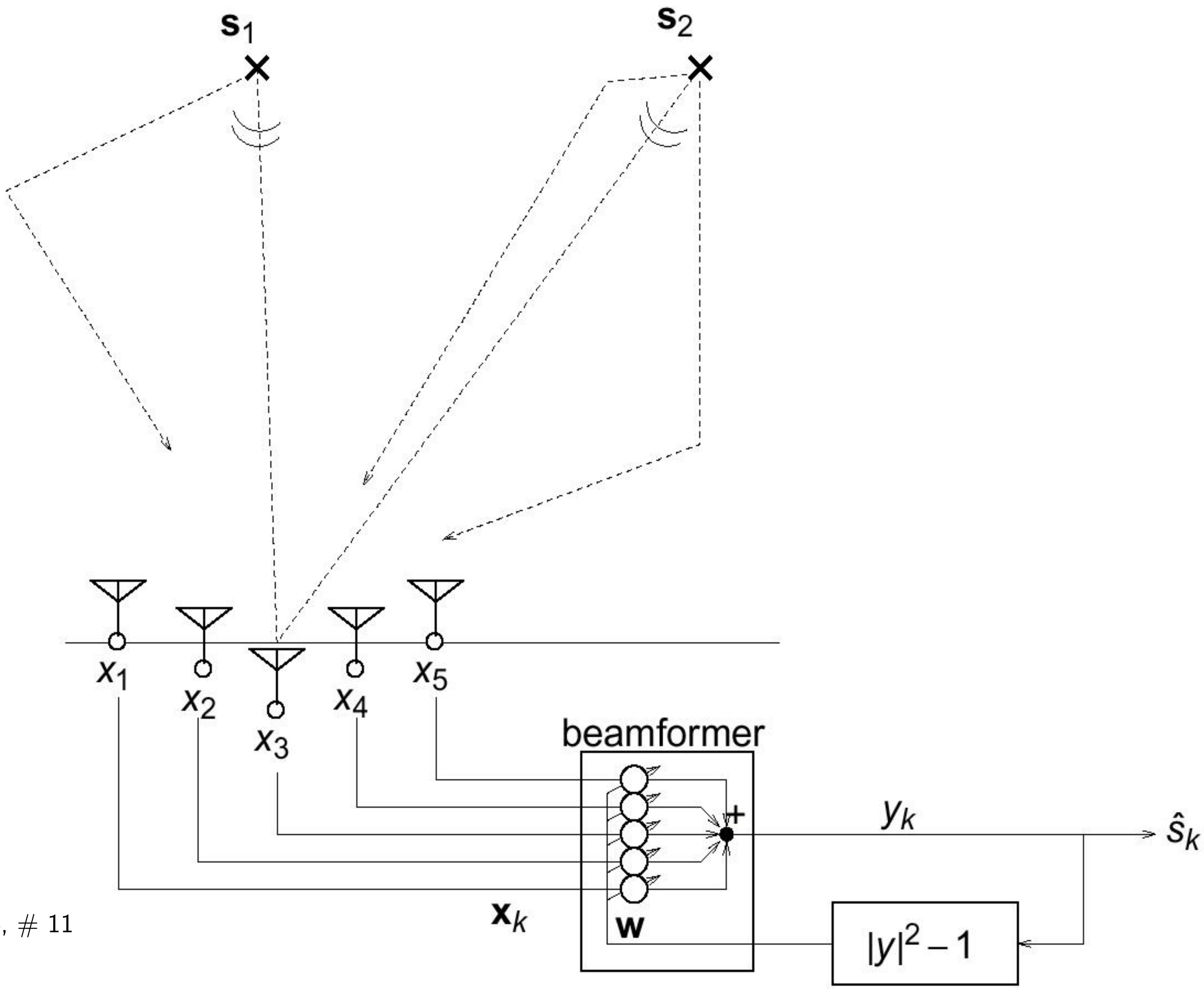
be a beamformer. Output of beamforming:

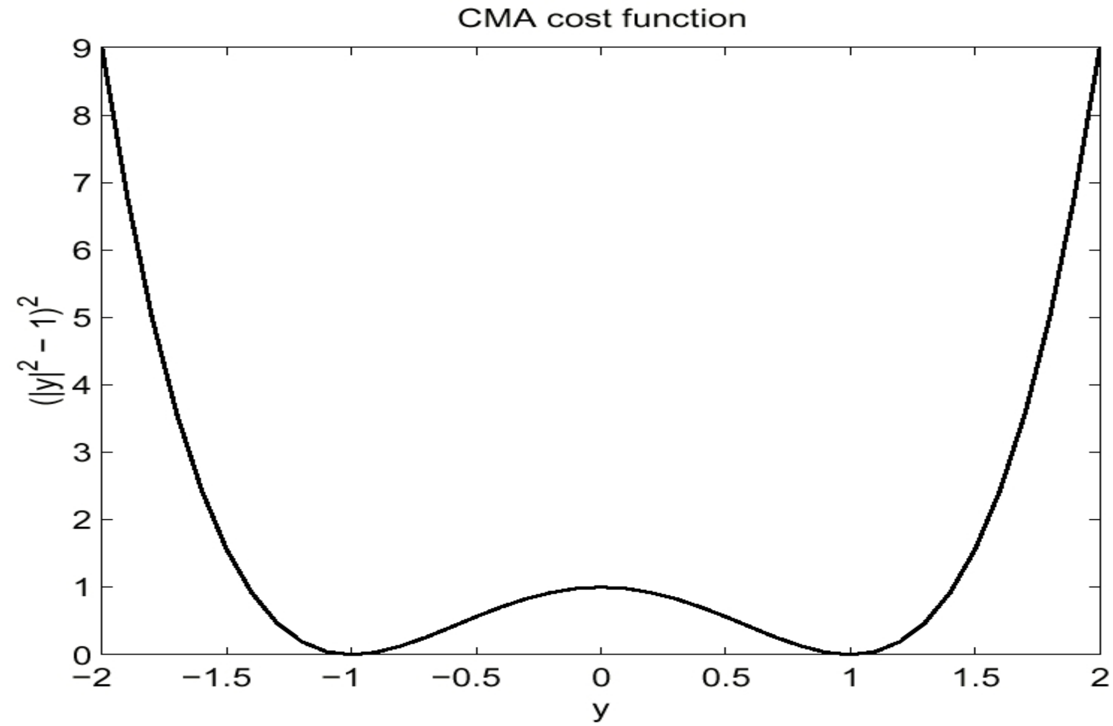
$$y_k = \boldsymbol{w}^H \boldsymbol{x}_k = [w_1^* \; w_2^*] \begin{bmatrix} x_{1,k} \\ x_{2,k} \end{bmatrix}.$$

Constant modulus property: $|s_{1,k}| = |s_{2,k}| = 1$ for all k.

Possible optimization problem:

$$\min J(\boldsymbol{w}) \quad \text{where} \quad J(\boldsymbol{w}) = \mathrm{E}\,[(|y_k|^2 - 1)^2].$$


$\mathbf{s}_1$
$\mathbf{s}_2$
x_1
x_2
x_3
x_4
x_5
beamformer
+
y_k
$\hat{s}_k$
$\mathbf{x}_k$
$\mathbf{w}$
$|y|^2 - 1$

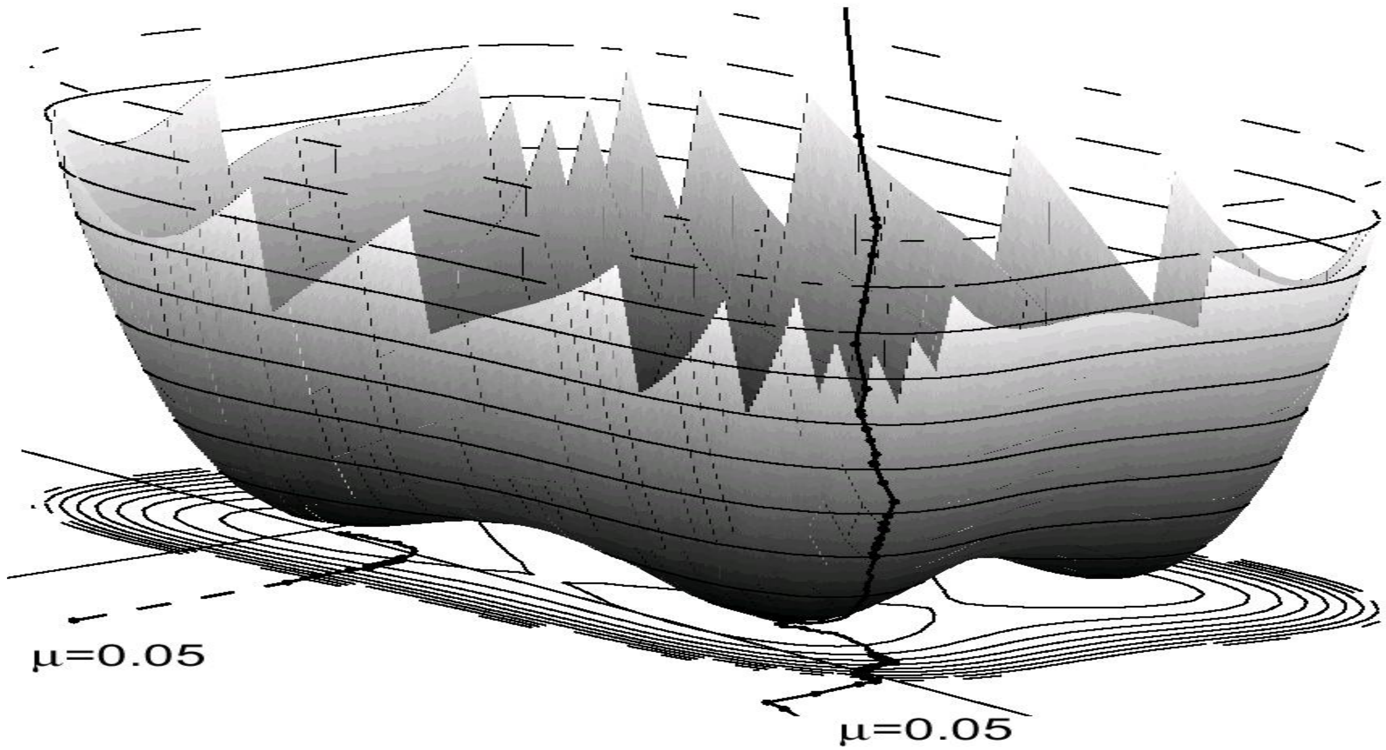


$$(|y_k|^2 - 1)^2$$

The CMA cost function as a function of y (for simplicity, y is taken to be *real* here).

No unique minimum! Indeed, if $y_k = \boldsymbol{w}^H \boldsymbol{x}_k$ is CM, then another beamformer is $\alpha \boldsymbol{w}$, for any scalar α that satisfies $|\alpha| = 1$.

2 (real-valued) sources, 2 antennas



Iterative Optimization

Cost function:

$$J(\boldsymbol{w}) = \mathrm{E}\,[(|y_k|^2 - 1)^2], \quad y_k = \boldsymbol{w}^H \boldsymbol{x}_k.$$

Stochastic gradient method: $\boldsymbol{w}_{k+1} = \boldsymbol{w}_k - \mu[\nabla J(\boldsymbol{w}_k)]^*$, where μ is step size, $\mu > 0$.

Derivative: Use $|y_k|^2 = y_k y_k^* = \boldsymbol{w}^H \boldsymbol{x}\boldsymbol{x}^H \boldsymbol{w}$.

$$\begin{aligned}
\nabla J &= 2\mathrm{E}\,\{(|y_k|^2 - 1)\cdot \nabla(\boldsymbol{w}^H \boldsymbol{x}_k \boldsymbol{x}_k^H \boldsymbol{w})\} \\
&= 2\mathrm{E}\,\{(|y_k|^2 - 1)\cdot (\boldsymbol{x}_k \boldsymbol{x}_k^H \boldsymbol{w})^*\} \\
&= 2\mathrm{E}\,\{(|y_k|^2 - 1)\boldsymbol{x}_k^* y_k\}
\end{aligned}$$

Algorithm CMA(2,2):

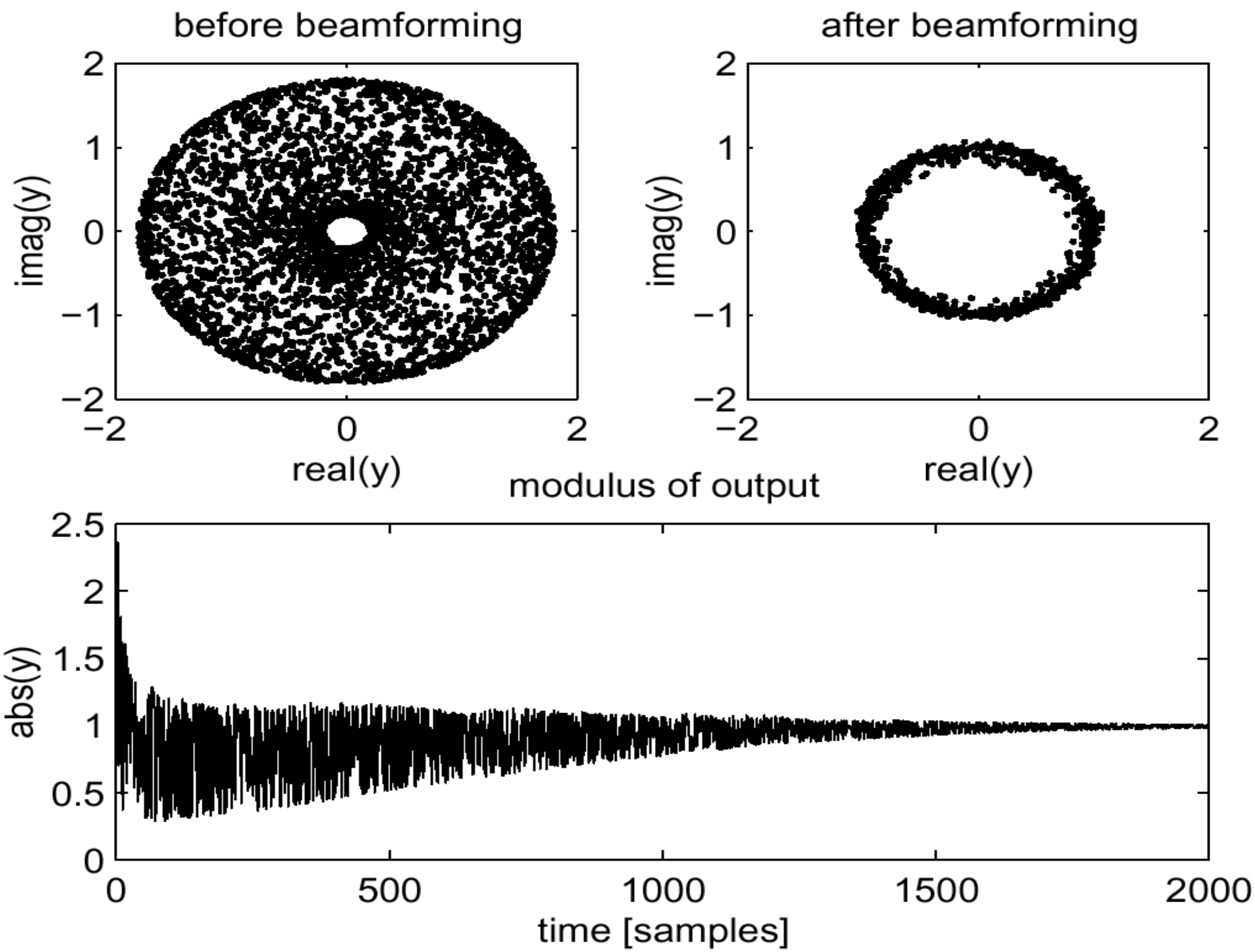
$$
\begin{aligned}
y_k &= \boldsymbol{w}_k^H \boldsymbol{x}_k \\
\boldsymbol{w}_{k+1} &= \boldsymbol{w}_k - \mu \boldsymbol{x}_k (|y_k|^2 - 1) y_k^*.
\end{aligned}
$$

Advantages:

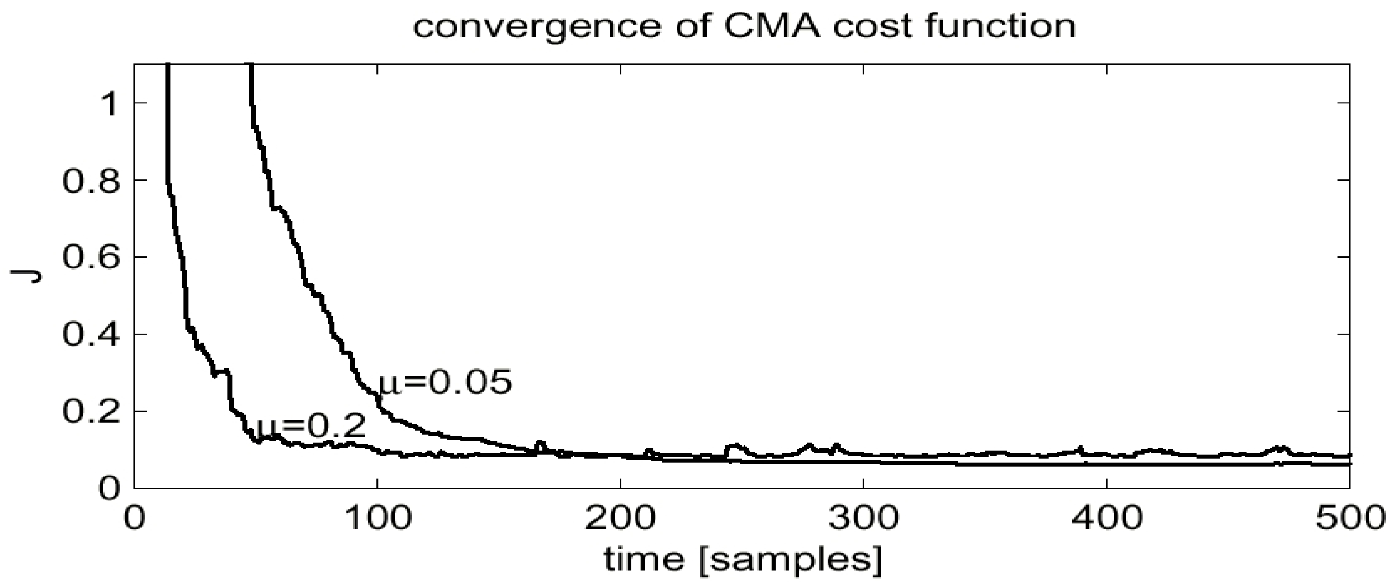
- The algorithm is extremely simple to implement
- Adaptive tracking of sources
- Converges to minima close to the Wiener beamformers (for each source)

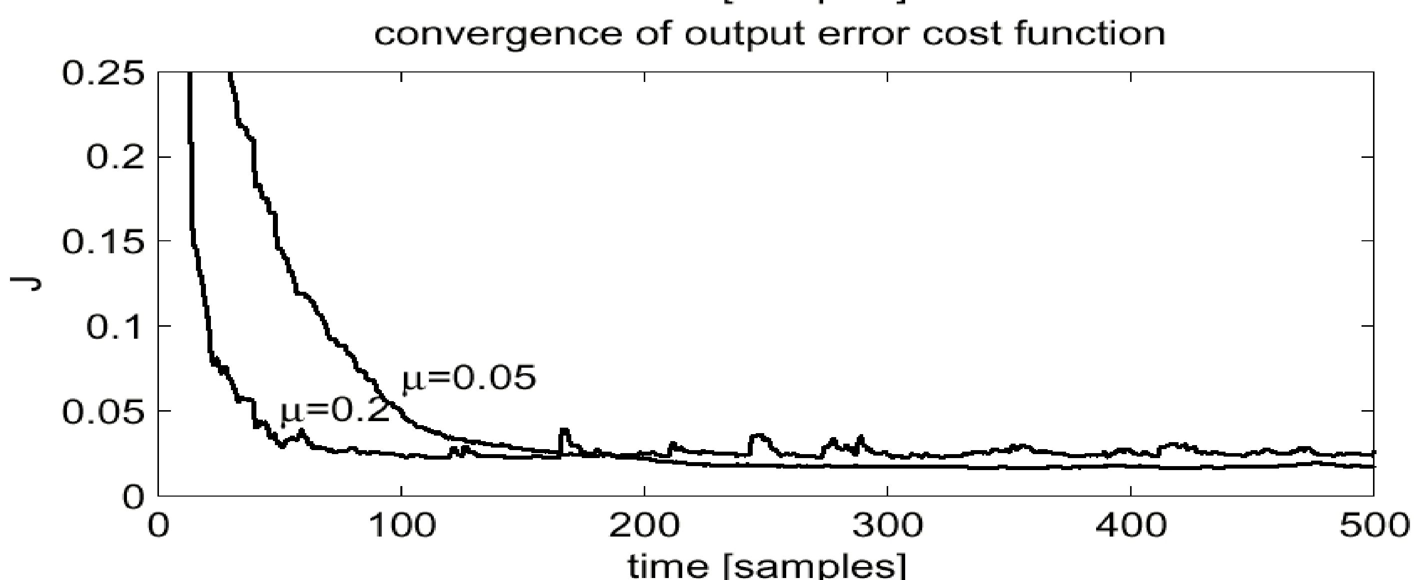
Disadvantages:

- Noisy and slow
- Step size μ should be small, else unstable
- Only one source is recovered (which one?)
- Possible convergence to local minimum (with finite data)

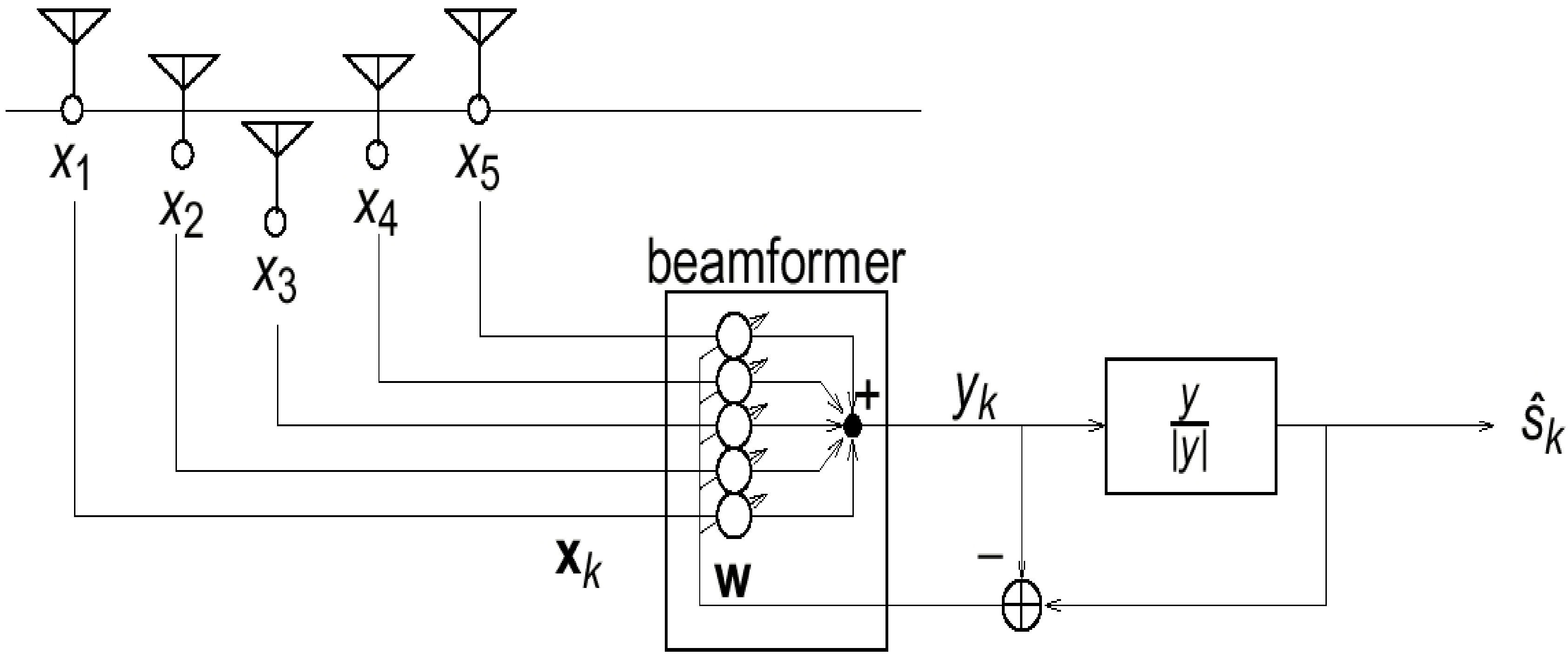


before beamforming
after beamforming
imag(y)
real(y)
2
1
0
−1
−2
−2
0
2
modulus of output
abs(y)
2.5
2
1.5
1
0.5
0
500
1000
1500
2000
time [samples]


convergence of CMA cost function
J
1
0.8
0.6
0.4
0.2
0
0
100
200
300
400
500
time [samples]
μ=0.05
μ=0.2


convergence of output error cost function
J
0.25
0.2
0.15
0.1
0.05
0
0
100
200
300
400
500
time [samples]
μ=0.05
μ=0.2

Other CMAs



Alternative cost function: CMA(1,2)

$$J(\boldsymbol{w}) = \mathrm{E}\left[(|y_k| - 1)^2\right] = \mathrm{E}\left[(|\boldsymbol{w}^H \boldsymbol{x}_k| - 1)^2\right].$$

Corresponding CMA iteration:

$$
\begin{aligned}
y_k &= \boldsymbol{w}_k^H \boldsymbol{x}_k \\
\epsilon_k &= \frac{y_k}{|y_k|} - y_k \\
\boldsymbol{w}_{k+1} &= \boldsymbol{w}_k + \mu \boldsymbol{x}_k \epsilon_k^*.
\end{aligned}
$$

Similar to LMS, with update error $\frac{y_k}{|y_k|} - y_k$. The desired signal is estimated by $\frac{y_k}{|y_k|}$.

Other CMAs (cont.)

- **Normalized CMA** (NCMA; μ becomes scaling independent)

$$\boldsymbol{w}_{k+1} = \boldsymbol{w}_k + \frac{\mu}{\|\boldsymbol{x}_k\|^2} \boldsymbol{x}_k \epsilon_k^*.$$

- **Orthogonal CMA** (OCMA): whiten using data covariance R

$$\boldsymbol{w}_{k+1} = \boldsymbol{w}_k + \mu R_k^{-1} \boldsymbol{x}_k \epsilon_k^*.$$

- **Least squares CMA** (LSCMA): block update, trying to optimize iteratively

$$\min_{\boldsymbol{w}} \|\widehat{\boldsymbol{s}}^H - \boldsymbol{w}^H X\|^2$$

where $X = [\boldsymbol{x}_1 \, \boldsymbol{x}_2 \cdots \boldsymbol{x}_T]$ and $\widehat{\boldsymbol{s}}^H$ is the best blind estimate at step k of the complete source vector (at all time points $t = 1, 2, \ldots, T$)

$$\widehat{\boldsymbol{s}}^H = \left[\frac{y_1}{|y_1|}, \frac{y_2}{|y_2|}, \ldots, \frac{y_T}{|y_T|}\right],$$

where

$$y_t = \boldsymbol{w}_k^H \boldsymbol{x}_t, \quad t = 1, 2, \ldots, K.$$

and

$$\boldsymbol{w}_k^H = \widehat{\boldsymbol{s}}^H X^H (X X^H)^{-1}.$$