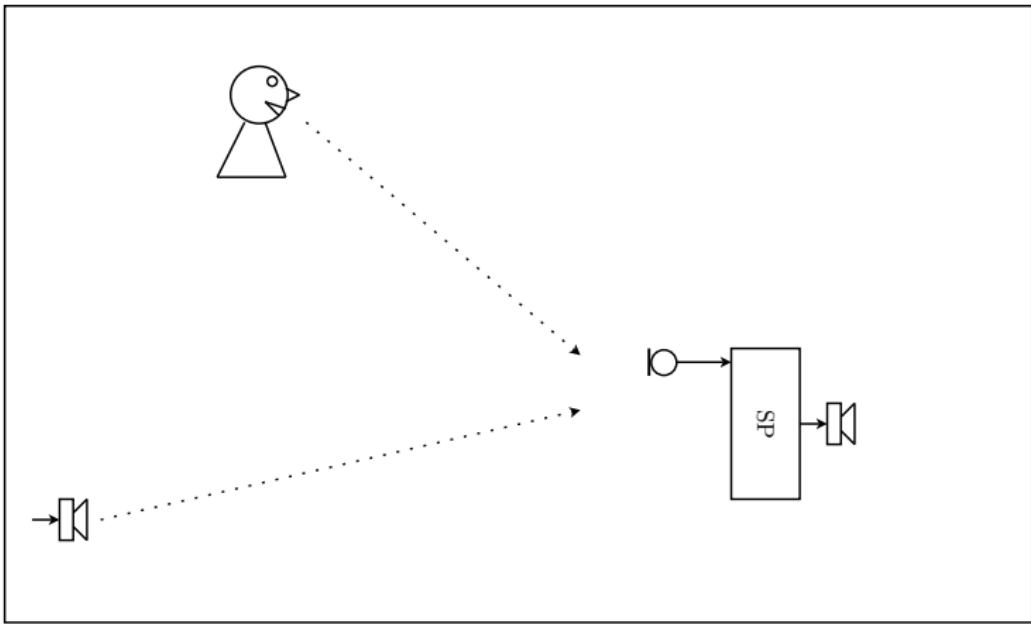


# Welcome to Adaptive Array Signal Processing (5SSC0)

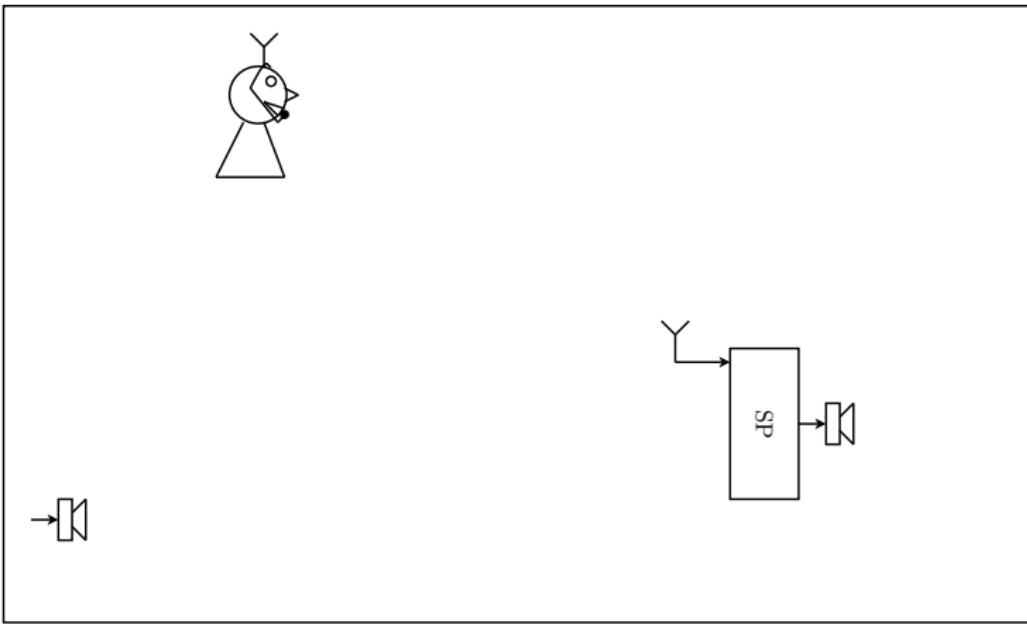
*by Ruud van Sloun (lecturer), Iris Huijben (instructor), Julian Merkofer (instructor), and Vincent van de Schaft (instructor).*

## Motivation AASP example: Skype

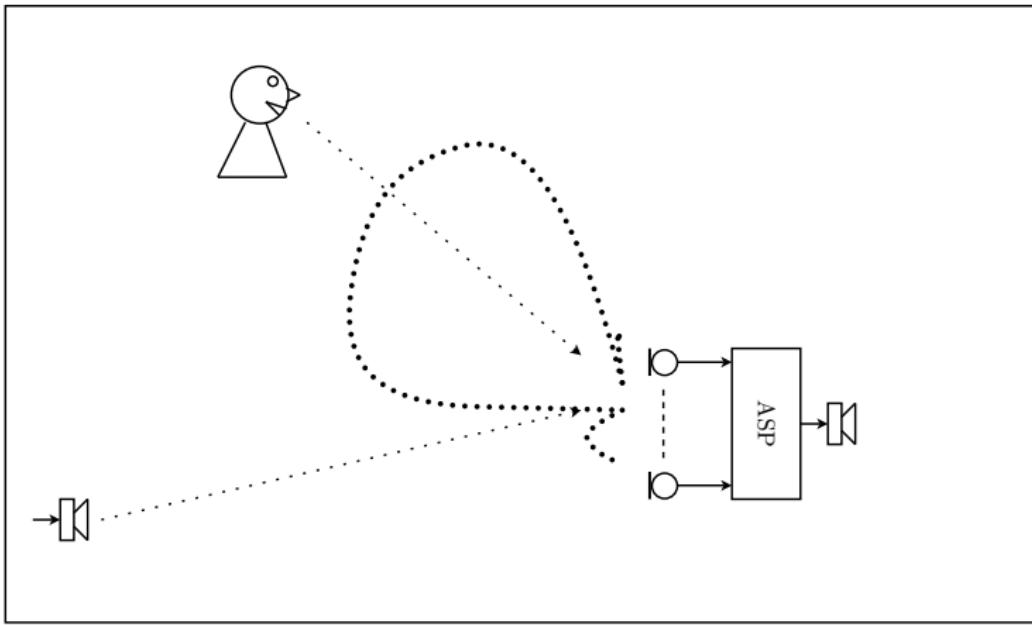
## Motivation AASP example: Skype



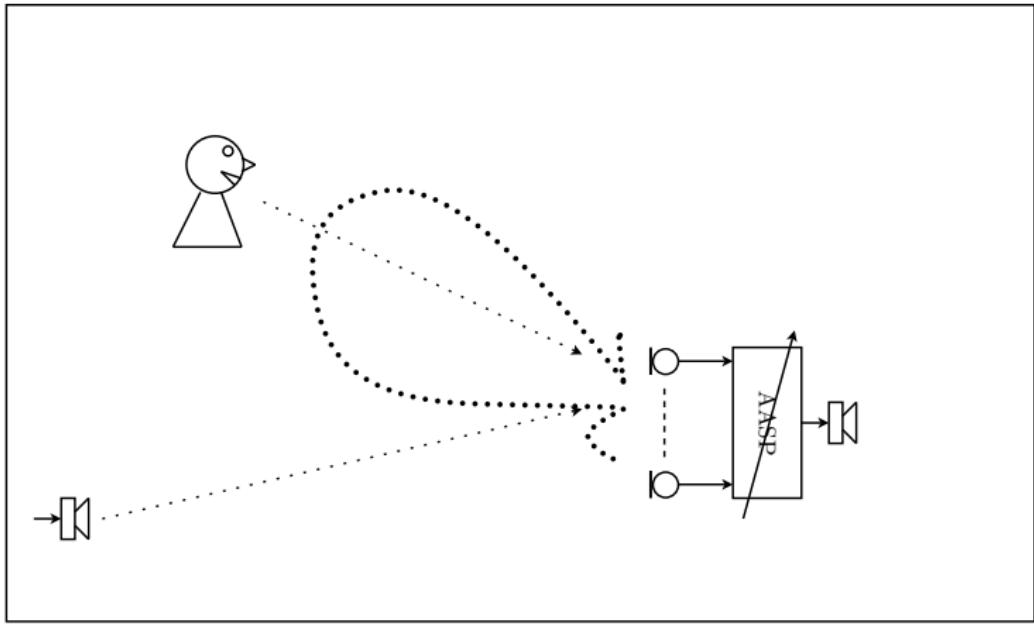
# Motivation AASP example: Skype



## Motivation AASP example: Skype

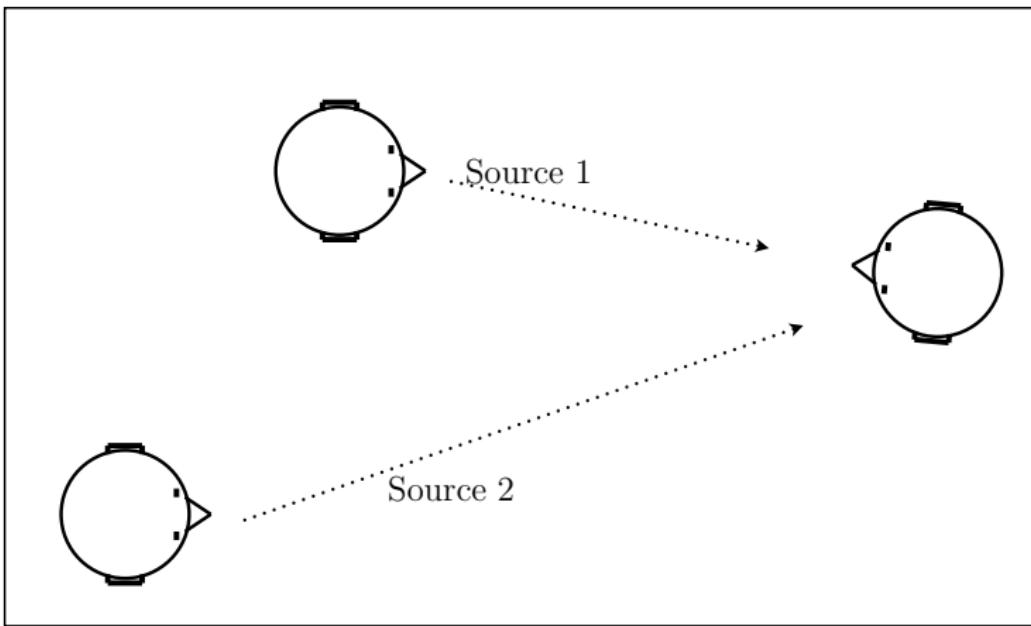


## Motivation AASP example: Skype

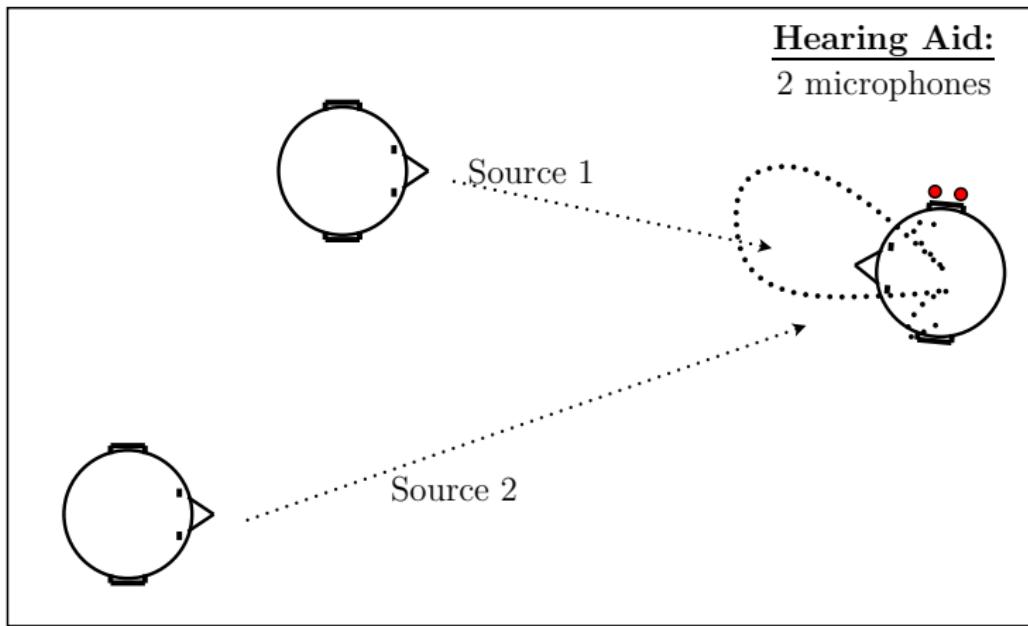


## Motivation AASP example: Hearing Aid

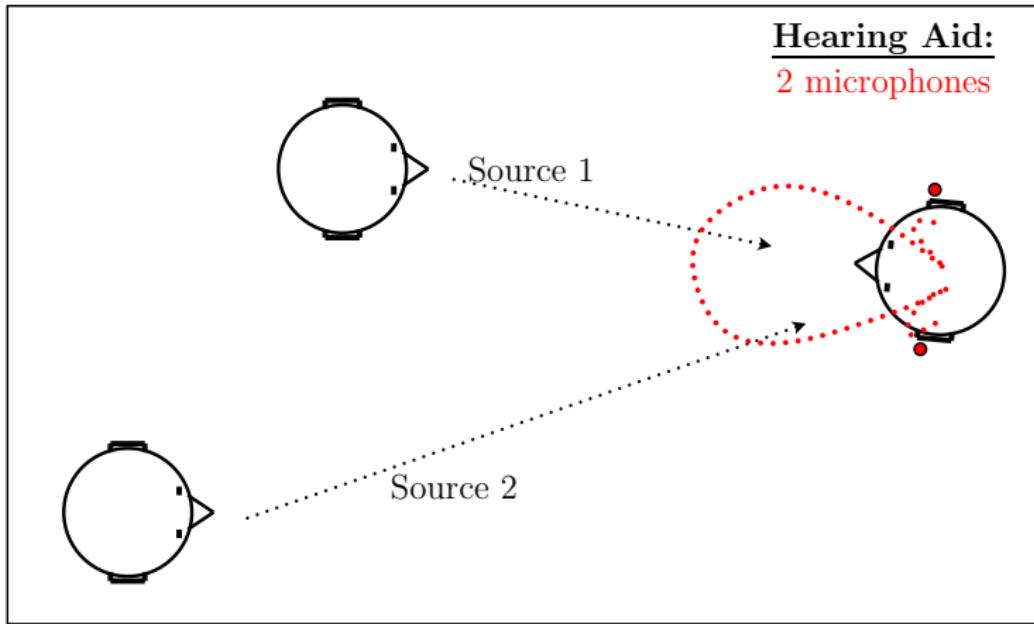
## Motivation AASP example: Hearing Aid



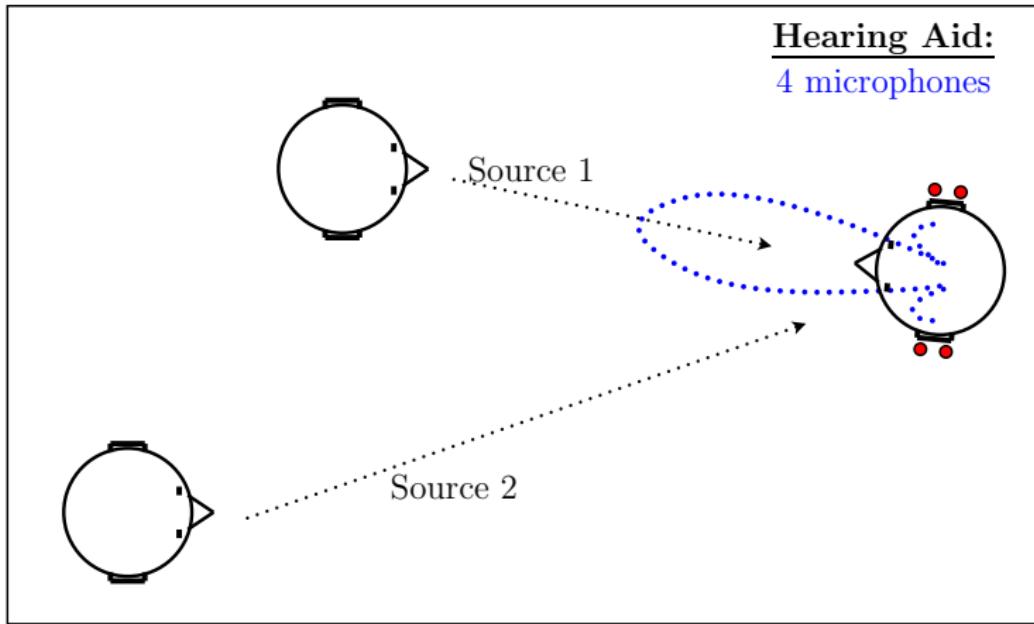
## Motivation AASP example: Hearing Aid



## Motivation AASP example: Hearing Aid



## Motivation AASP example: Hearing Aid



# Introduction

- **Main content:**

# Introduction

- Main content:

- ▶ Part 1A: **Adaptive Signal Processing** (*Single channel, FIR*)
- ▶ Part 1B: **Array Signal Processing (ASP)** (*including DOA*)
- ▶ Part 1C: **Adaptive Array Signal Processing (AASP)**
- ▶ Part 2A: **Compressive sensing: theory**
- ▶ Part 2B: **Compressive sensing: state of the art**

# Introduction

- Main content:
  - ▶ Part 1A: **Adaptive Signal Processing** (*Single channel, FIR*)
  - ▶ Part 1B: **Array Signal Processing (ASP)** (*including DOA*)
  - ▶ Part 1C: **Adaptive Array Signal Processing (AASP)**
  - ▶ Part 2A: **Compressive sensing: theory**
  - ▶ Part 2B: **Compressive sensing: state of the art**
- Assignments and oral exam:
  1. Groups of 2 students each

# Introduction

- **Main content:**
  - ▶ Part 1A: **Adaptive Signal Processing** (*Single channel, FIR*)
  - ▶ Part 1B: **Array Signal Processing (ASP)** (*including DOA*)
  - ▶ Part 1C: **Adaptive Array Signal Processing (AASP)**
  - ▶ Part 2A: **Compressive sensing: theory**
  - ▶ Part 2B: **Compressive sensing: state of the art**

- **Assignments and oral exam:**

1. Groups of 2 students each
2. Assignments 1A, 1B and 1C, 2:  
Fill in predefined documents

# Introduction

- **Main content:**

- ▶ Part 1A: **Adaptive Signal Processing** (*Single channel, FIR*)
- ▶ Part 1B: **Array Signal Processing (ASP)** (*including DOA*)
- ▶ Part 1C: **Adaptive Array Signal Processing (AASP)**
- ▶ Part 2A: **Compressive sensing: theory**
- ▶ Part 2B: **Compressive sensing: state of the art**

- **Assignments and oral exam:**

1. Groups of 2 students each
2. Assignments 1A, 1B and 1C, 2:  
Fill in predefined documents
3. All reports to be finalized during course

# Introduction

- **Main content:**

- ▶ Part 1A: **Adaptive Signal Processing** (*Single channel, FIR*)
- ▶ Part 1B: **Array Signal Processing (ASP)** (*including DOA*)
- ▶ Part 1C: **Adaptive Array Signal Processing (AASP)**
- ▶ Part 2A: **Compressive sensing: theory**
- ▶ Part 2B: **Compressive sensing: state of the art**

- **Assignments and oral exam:**

1. Groups of 2 students each
2. Assignments 1A, 1B and 1C, 2:  
Fill in predefined documents
3. All reports to be finalized during course
4. Oral exam: During exam week of Q3

# Introduction

- **Main content:**

- ▶ Part 1A: **Adaptive Signal Processing** (*Single channel, FIR*)
- ▶ Part 1B: **Array Signal Processing (ASP)** (*including DOA*)
- ▶ Part 1C: **Adaptive Array Signal Processing (AASP)**
- ▶ Part 2A: **Compressive sensing: theory**
- ▶ Part 2B: **Compressive sensing: state of the art**

- **Assignments and oral exam:**

1. Groups of 2 students each
2. Assignments 1A, 1B and 1C, 2:  
Fill in predefined documents
3. All reports to be finalized during course
4. Oral exam: During exam week of Q3 **To pass 5SSC0 ⇒ ORAL ≥ 5**

# Background material

## 1. Book:

Dimitris G. Manolakis, Vinay K. Ingle and Stephen M. Kogon,  
"Statistical and Adaptive Array Signal Processing: Spectral  
estimation, signal modeling, adaptive filtering and array  
processing"

- ▶ Optimum linear filters (Chapter 6)
- ▶ Adaptive filters (Chapter 10)
- ▶ Array processing (Chapter 11)

## 2. Book:

Duarte and Eldar, "Structured Compressed Sensing: From  
Theory to Applications", from Sec. I to II-C  
(<https://arxiv.org/pdf/1106.6224.pdf>)

# Background material

## 1. Book:

Dimitris G. Manolakis, Vinay K. Ingle and Stephen M. Kogon,  
"Statistical and Adaptive Array Signal Processing: Spectral  
estimation, signal modeling, adaptive filtering and array  
processing"

- ▶ Optimum linear filters (Chapter 6)
- ▶ Adaptive filters (Chapter 10)
- ▶ Array processing (Chapter 11)

## 2. Book:

Duarte and Eldar, "Structured Compressed Sensing: From  
Theory to Applications", from Sec. I to II-C  
(<https://arxiv.org/pdf/1106.6224.pdf>)

## 3. These slides

# Background material

## 1. Book:

Dimitris G. Manolakis, Vinay K. Ingle and Stephen M. Kogon,  
"Statistical and Adaptive Array Signal Processing: Spectral  
estimation, signal modeling, adaptive filtering and array  
processing"

- ▶ Optimum linear filters (Chapter 6)
- ▶ Adaptive filters (Chapter 10)
- ▶ Array processing (Chapter 11)

## 2. Book:

Duarte and Eldar, "Structured Compressed Sensing: From  
Theory to Applications", from Sec. I to II-C  
(<https://arxiv.org/pdf/1106.6224.pdf>)

## 3. These slides

## 4. Documents: Study guide/ Course organization.

# Background material

## 1. Book:

Dimitris G. Manolakis, Vinay K. Ingle and Stephen M. Kogon,  
"Statistical and Adaptive Array Signal Processing: Spectral  
estimation, signal modeling, adaptive filtering and array  
processing"

- ▶ Optimum linear filters (Chapter 6)
- ▶ Adaptive filters (Chapter 10)
- ▶ Array processing (Chapter 11)

## 2. Book:

Duarte and Eldar, "Structured Compressed Sensing: From  
Theory to Applications", from Sec. I to II-C  
(<https://arxiv.org/pdf/1106.6224.pdf>)

## 3. These slides

4. Documents: Study guide/ Course organization.
5. Necessary Matlab code

# Background material

## 1. Book:

Dimitris G. Manolakis, Vinay K. Ingle and Stephen M. Kogon,  
"Statistical and Adaptive Array Signal Processing: Spectral  
estimation, signal modeling, adaptive filtering and array  
processing"

- ▶ Optimum linear filters (Chapter 6)
- ▶ Adaptive filters (Chapter 10)
- ▶ Array processing (Chapter 11)

## 2. Book:

Duarte and Eldar, "Structured Compressed Sensing: From  
Theory to Applications", from Sec. I to II-C  
(<https://arxiv.org/pdf/1106.6224.pdf>)

## 3. These slides

4. Documents: Study guide/ Course organization.
5. Necessary Matlab code

All relevant material (except book) available via Canvas (Modules):

# Background material

## 1. Book:

Dimitris G. Manolakis, Vinay K. Ingle and Stephen M. Kogon,  
"Statistical and Adaptive Array Signal Processing: Spectral  
estimation, signal modeling, adaptive filtering and array  
processing"

- ▶ Optimum linear filters (Chapter 6)
- ▶ Adaptive filters (Chapter 10)
- ▶ Array processing (Chapter 11)

## 2. Book:

Duarte and Eldar, "Structured Compressed Sensing: From  
Theory to Applications", from Sec. I to II-C  
(<https://arxiv.org/pdf/1106.6224.pdf>)

## 3. These slides

4. Documents: Study guide/ Course organization.
5. Necessary Matlab code

All relevant material (except book) available via Canvas (Modules):

Register for a group in Canvas

## Deadlines and credits

| Code  | Deadline / Date    | Credits |
|-------|--------------------|---------|
| 1A    | February 20, 09:00 | 10      |
| 1B    | March 6, 09:00     | 10      |
| 1C    | March 20, 09:00    | 10      |
| 2     | April 10, 09:00    | 20      |
| Oral  | tbd (April 10-21)  | 50      |
| Total |                    | 100     |

## Deadlines and credits

| Code  | Deadline / Date    | Credits |
|-------|--------------------|---------|
| 1A    | February 20, 09:00 | 10      |
| 1B    | March 6, 09:00     | 10      |
| 1C    | March 20, 09:00    | 10      |
| 2     | April 10, 09:00    | 20      |
| Oral  | tbd (April 10-21)  | 50      |
| Total |                    | 100     |

To pass 5SSC0  $\Rightarrow$  ORAL  $\geq 5$

# **Adaptive Signal Processing**

## **(Part IA)**

# Content part I

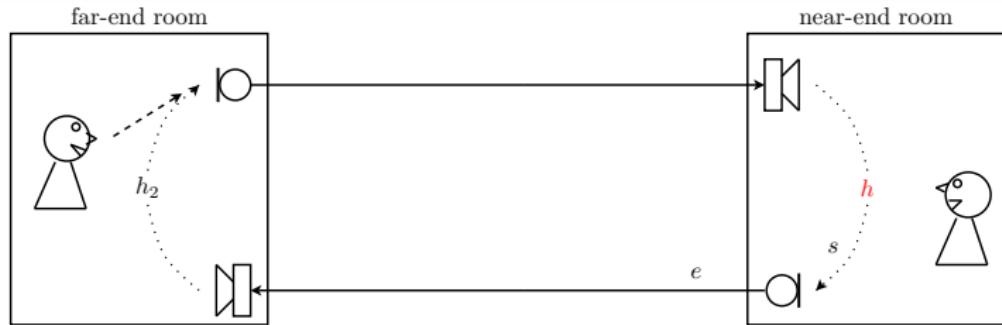
*Focus on single channel adaptive algorithms using FIR structures*

# Content part I

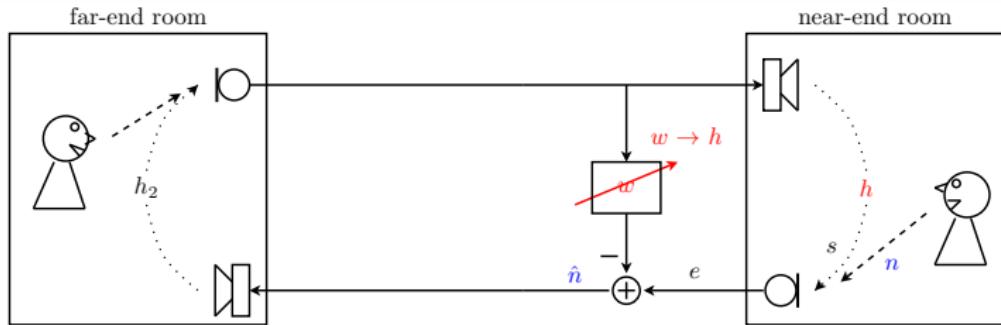
*Focus on single channel adaptive algorithms using FIR structures*

- ▶ Applications
- ▶ Minimum Mean Squared Error (MMSE)
- ▶ Constrained MMSE
- ▶ Least Squares (LS)
- ▶ Steepest Descent Algorithm (SGD)
- ▶ LMS variants: (Complex) (N)LMS, Constrained LMS
- ▶ Newton algorithm
- ▶ Recursive Least Squares (RLS)
- ▶ Frequency Domain Adaptive Filter (FDAF)
- ▶ Summary

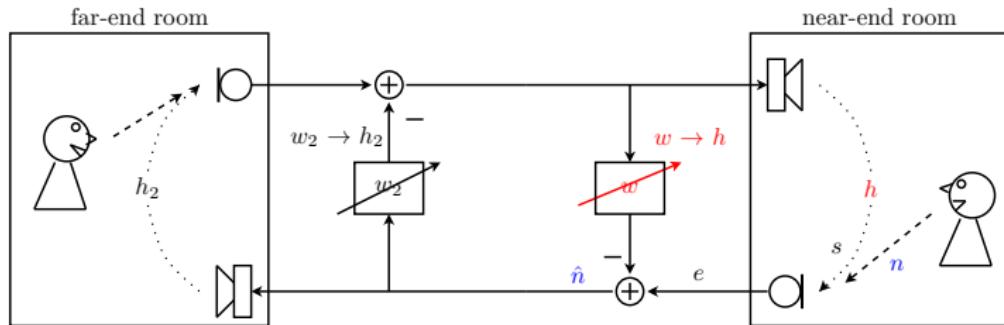
# Applications: Acoustic Echo Cancellation



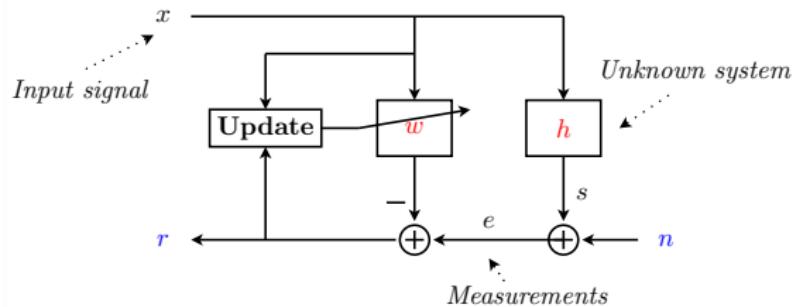
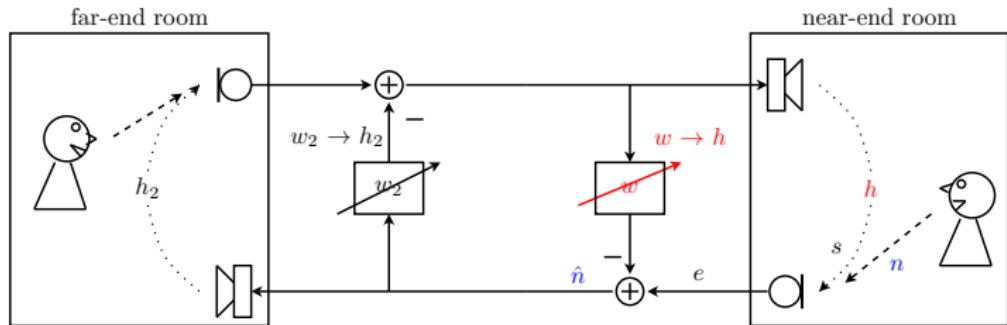
# Applications: Acoustic Echo Cancellation



# Applications: Acoustic Echo Cancellation

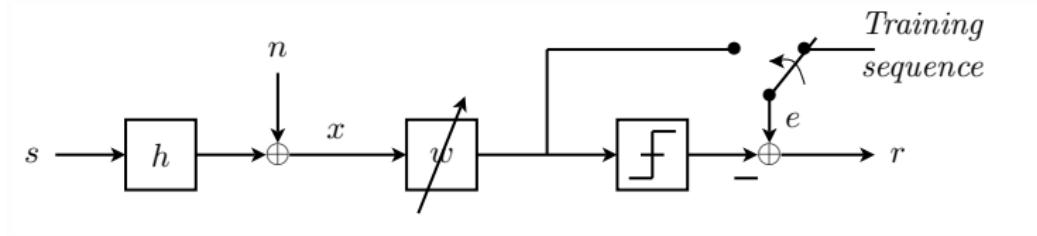


# Applications: Acoustic Echo Cancellation

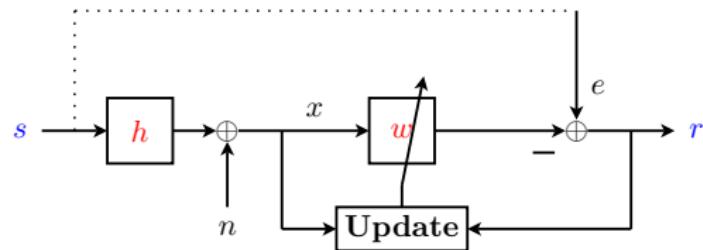
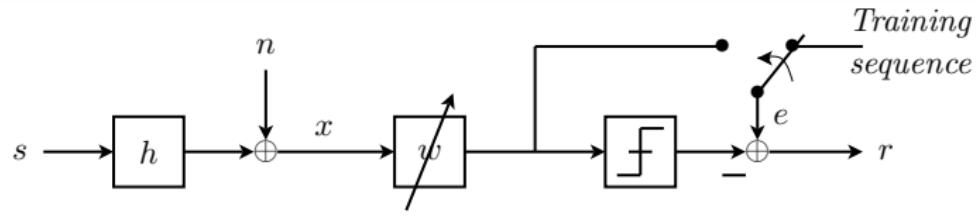


**System Identification:**  $w \rightarrow h \Rightarrow r \rightarrow n$

## Applications: Equalization

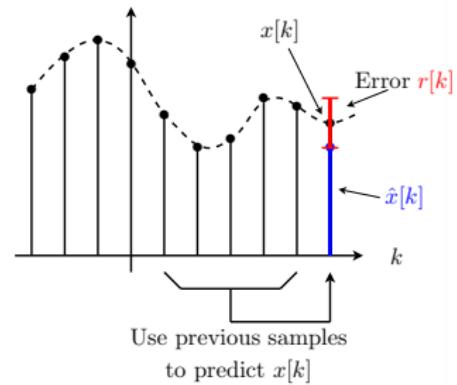


# Applications: Equalization

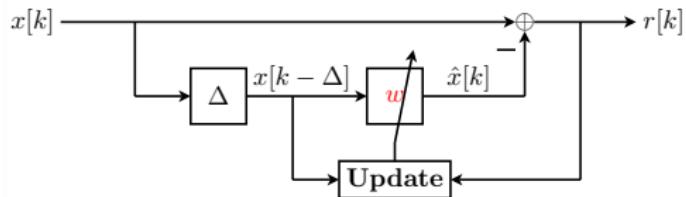
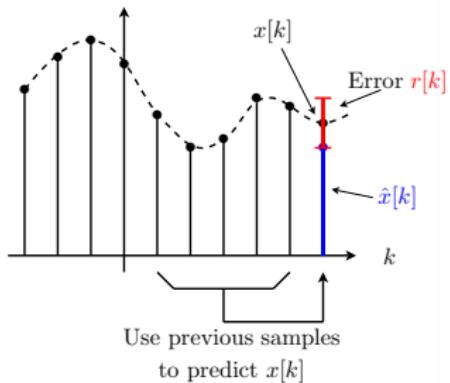


Signal correction/ Inverse modelling:  $w \rightarrow h^{-1} \Rightarrow r \rightarrow s$

# Applications: Signal prediction

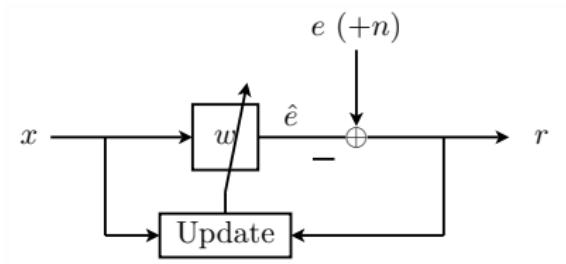


# Applications: Signal prediction

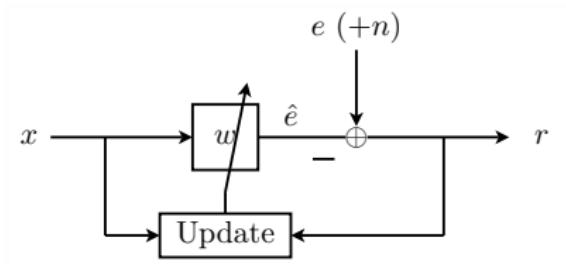


Signal prediction: Predict  $x[k]$  from  $x[k - \Delta]$

# General Adaptive Filter model



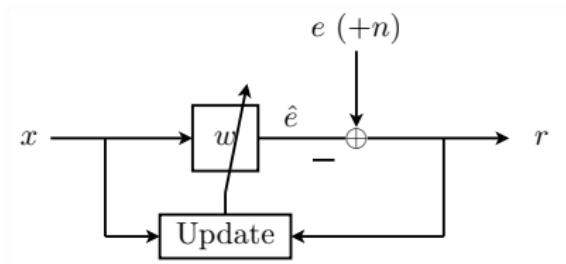
# General Adaptive Filter model



Notes:

- ▶ Signals  $x$  and  $e$  correlated

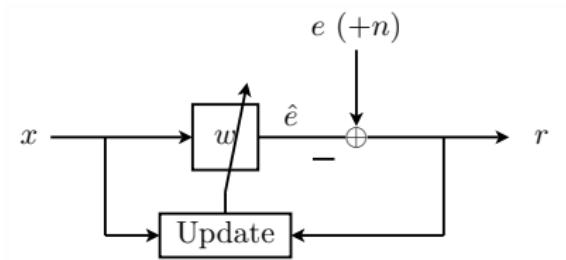
# General Adaptive Filter model



Notes:

- ▶ Signals  $x$  and  $e$  correlated
- ▶ "Noise"  $n$  not correlated with other signals

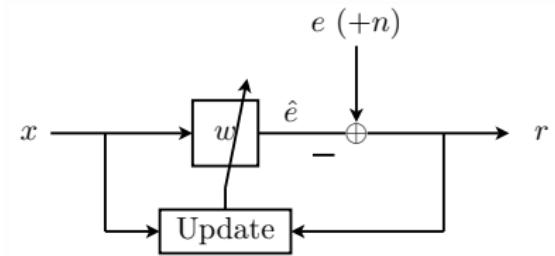
# General Adaptive Filter model



## Notes:

- ▶ Signals  $x$  and  $e$  correlated
- ▶ "Noise"  $n$  not correlated with other signals
- ▶ Pragmatic choices:
  - ▶ All signals in average zero
  - ▶ Filter  $w$ : FIR

# General Adaptive Filter model

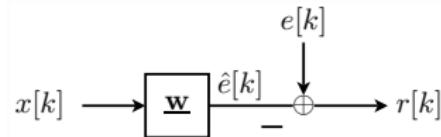


## Notes:

- ▶ Signals  $x$  and  $e$  correlated
- ▶ "Noise"  $n$  not correlated with other signals
- ▶ Pragmatic choices:
  - ▶ All signals in average zero
  - ▶ Filter  $w$ : FIR
- ▶ Calculation of weight of filter  $w$ :
  - ▶ Use quadratic cost function:  $J = f(r^2)$
  - ▶ **First fixed weights** (MMSE, LS), then adaptive

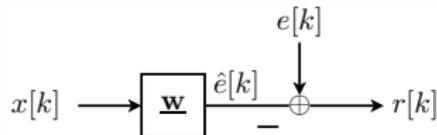
# Fixed weights: MMSE

## General Minimum Mean Squared Error (MMSE) model:



# Fixed weights: MMSE

## General Minimum Mean Squared Error (MMSE) model:

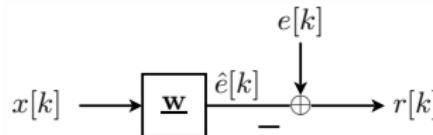


### Goal:

Given  $N$  samples  $\underline{x}[k] = (x[k], x[k - 1], \dots, x[k - N + 1])^t$  calculate coefficients fixed filter  $\underline{\mathbf{w}} = (w_0, w_1, \dots, w_{N-1})^t$  such that Mean Squared Error (MSE)  
 $J = E\{r^2[k]\} = E\{(e[k] - \hat{e}[k])^2\}$  is minimized.

# Fixed weights: MMSE

## General Minimum Mean Squared Error (MMSE) model:



### Goal:

Given  $N$  samples  $\underline{x}[k] = (x[k], x[k - 1], \dots, x[k - N + 1])^t$   
calculate coefficients fixed filter  $\underline{w} = (w_0, w_1, \dots, w_{N-1})^t$  such  
that Mean Squared Error (MSE)  
 $J = E\{r^2[k]\} = E\{(e[k] - \hat{e}[k])^2\}$  is minimized.

### MMSE Optimization problem:

Given FIR samples  $x[k - i]$  for  $i = 0, 1, \dots, N - 1$

$$\underline{w}_o = \arg \min_{\underline{w}} (E\{r^2[k]\})$$

## Fixed weights: MMSE

$$\begin{aligned} J &= E\{(e[k] - \underline{w}^t \cdot \underline{x}[k]) \cdot (e[k] - \underline{x}^t[k] \cdot \underline{w})\} \\ &= E\{e^2[k]\} - \underline{w}^t E\{\underline{x}[k]e[k]\} - E\{e[k]\underline{x}^t[k]\}\underline{w} + \underline{w}^t E\{\underline{x}[k]\underline{x}^t[k]\}\underline{w} \end{aligned}$$

## Fixed weights: MMSE

$$\begin{aligned} J &= E\{(e[k] - \underline{w}^t \cdot \underline{x}[k]) \cdot (e[k] - \underline{x}^t[k] \cdot \underline{w})\} \\ &= E\{e^2[k]\} - \underline{w}^t E\{\underline{x}[k]e[k]\} - E\{e[k]\underline{x}^t[k]\}\underline{w} + \underline{w}^t E\{\underline{x}[k]\underline{x}^t[k]\}\underline{w} \end{aligned}$$

$$\Rightarrow J = E\{e^2[k]\} - \underline{w}^t \underline{r}_{ex} - \underline{r}_{ex}^t \underline{w} + \underline{w}^t \mathbf{R}_x \underline{w}$$

## Fixed weights: MMSE

$$\begin{aligned} J &= E\{(e[k] - \underline{w}^t \cdot \underline{x}[k]) \cdot (e[k] - \underline{x}^t[k] \cdot \underline{w})\} \\ &= E\{e^2[k]\} - \underline{w}^t E\{\underline{x}[k]e[k]\} - E\{e[k]\underline{x}^t[k]\}\underline{w} + \underline{w}^t E\{\underline{x}[k]\underline{x}^t[k]\}\underline{w} \end{aligned}$$

$$\Rightarrow J = E\{e^2[k]\} - \underline{w}^t \underline{r}_{ex} - \underline{r}_{ex}^t \underline{w} + \underline{w}^t R_x \underline{w}$$

with cross correlation  $\rho_{ex}[\tau] = E\{e[k]x[k - \tau]\}$ :

$$\underline{r}_{ex} = E\{e[k]\underline{x}[k]\} = (\rho_{ex}[0], \rho_{ex}[1], \dots, \rho_{ex}[N-1])^t$$

## Fixed weights: MMSE

$$\begin{aligned} J &= E\{(e[k] - \underline{w}^t \cdot \underline{x}[k]) \cdot (e[k] - \underline{x}^t[k] \cdot \underline{w})\} \\ &= E\{e^2[k]\} - \underline{w}^t E\{\underline{x}[k]e[k]\} - E\{e[k]\underline{x}^t[k]\}\underline{w} + \underline{w}^t E\{\underline{x}[k]\underline{x}^t[k]\}\underline{w} \end{aligned}$$

$$\Rightarrow J = E\{e^2[k]\} - \underline{w}^t \underline{r}_{ex} - \underline{r}_{ex}^t \underline{w} + \underline{w}^t R_x \underline{w}$$

with cross correlation  $\rho_{ex}[\tau] = E\{e[k]x[k - \tau]\}$ :

$$\underline{r}_{ex} = E\{e[k]\underline{x}[k]\} = (\rho_{ex}[0], \rho_{ex}[1], \dots, \rho_{ex}[N-1])^t$$

and autocorrelation:  $\rho_x[\tau] = E\{x[k]x[k - \tau]\} = rho_x[-\tau]$

$$R_x = E\{\underline{x}[k]\underline{x}^t[k]\} = \begin{pmatrix} \rho_x[0] & \rho_x[1] & \cdots & \rho_x[N-1] \\ \rho_x[1] & \rho_x[0] & \cdots & \rho_x[N-2] \\ \vdots & \vdots & \ddots & \vdots \\ \rho_x[N-1] & \rho_x[N-2] & \cdots & \rho_x[0] \end{pmatrix}$$

## Fixed weights: MMSE

$$J = E\{e^2[k]\} - \underline{w}^t \underline{r}_{ex} - \underline{r}_{ex}^t \underline{w} + \underline{w}^t \mathbf{R}_x \underline{w}$$

## Fixed weights: MMSE

$$J = E\{e^2[k]\} - \underline{w}^t \underline{r}_{ex} - \underline{r}_{ex}^t \underline{w} + \underline{w}^t \mathbf{R}_x \underline{w}$$

$$\Rightarrow \text{Optimum: } \underline{\nabla} = \frac{dJ}{dw} = -2(\underline{r}_{ex} - \mathbf{R}_x \underline{w}) = \underline{0}$$

## Fixed weights: MMSE

$$J = E\{e^2[k]\} - \underline{w}^t \underline{r}_{ex} - \underline{r}_{ex}^t \underline{w} + \underline{w}^t \mathbf{R}_x \underline{w}$$

$$\Rightarrow \text{Optimum: } \underline{\nabla} = \frac{dJ}{dw} = -2(\underline{r}_{ex} - \mathbf{R}_x \underline{w}) = \underline{0}$$

$\Rightarrow$  **Normal Equations**

$$\boxed{\mathbf{R}_x \cdot \underline{w} = \underline{r}_{ex}}$$

## Fixed weights: MMSE

$$J = E\{e^2[k]\} - \underline{w}^t \underline{r}_{ex} - \underline{r}_{ex}^t \underline{w} + \underline{w}^t \mathbf{R}_x \underline{w}$$

$$\Rightarrow \text{Optimum: } \bigtriangledown = \frac{dJ}{dw} = -2(\underline{r}_{ex} - \mathbf{R}_x \underline{w}) = 0$$

$\Rightarrow$  **Normal Equations**

$$\mathbf{R}_x \cdot \underline{w} = \underline{r}_{ex}$$

$\Rightarrow$  **Wiener filter**

$$\underline{w}_o = \mathbf{R}_x^{-1} \cdot \underline{r}_{ex}$$

## Fixed weights: MMSE

$$J = E\{e^2[k]\} - \underline{w}^t \underline{r}_{ex} - \underline{r}_{ex}^t \underline{w} + \underline{w}^t \mathbf{R}_x \underline{w}$$

$$\Rightarrow \text{Optimum: } \underline{\triangledown} = \frac{dJ}{dw} = -2(\underline{r}_{ex} - \mathbf{R}_x \underline{w}) = \underline{0}$$

$\Rightarrow$  **Normal Equations**

$$\mathbf{R}_x \cdot \underline{w} = \underline{r}_{ex}$$

$\Rightarrow$  **Wiener filter**

$$\underline{w}_o = \mathbf{R}_x^{-1} \cdot \underline{r}_{ex}$$

**General expression:**  $J = J_{min} + (\underline{w} - \underline{w}_o)^t \cdot \mathbf{R}_x \cdot (\underline{w} - \underline{w}_o)$

## Fixed weights: MMSE

$$J = E\{e^2[k]\} - \underline{w}^t \underline{r}_{ex} - \underline{r}_{ex}^t \underline{w} + \underline{w}^t \mathbf{R}_x \underline{w}$$

$$\Rightarrow \text{Optimum: } \underline{\triangledown} = \frac{dJ}{dw} = -2(\underline{r}_{ex} - \mathbf{R}_x \underline{w}) = \underline{0}$$

$\Rightarrow$  **Normal Equations**

$$\mathbf{R}_x \cdot \underline{w} = \underline{r}_{ex}$$

$\Rightarrow$  **Wiener filter**

$$\underline{w}_o = \mathbf{R}_x^{-1} \cdot \underline{r}_{ex}$$

**General expression:**  $J = J_{min} + (\underline{w} - \underline{w}_o)^t \cdot \mathbf{R}_x \cdot (\underline{w} - \underline{w}_o)$

$$J_{min} = J_{\underline{w}=\underline{w}_o} = E\{r^2[k]\} = E\{r[k] \cdot e[k]\} = E\{e^2[k]\} - \underline{r}_{ex}^t \mathbf{R}_x^{-1} \underline{r}_{ex}$$

## Fixed weights: MMSE

$$J = E\{e^2[k]\} - \underline{w}^t \underline{r}_{ex} - \underline{r}_{ex}^t \underline{w} + \underline{w}^t \mathbf{R}_x \underline{w}$$

$$\Rightarrow \text{Optimum: } \underline{\nabla} = \frac{dJ}{d\underline{w}} = -2(\underline{r}_{ex} - \mathbf{R}_x \underline{w}) = \underline{0}$$

$\Rightarrow$  **Normal Equations**

$$\mathbf{R}_x \cdot \underline{w} = \underline{r}_{ex}$$

$\Rightarrow$  **Wiener filter**

$$\underline{w}_o = \mathbf{R}_x^{-1} \cdot \underline{r}_{ex}$$

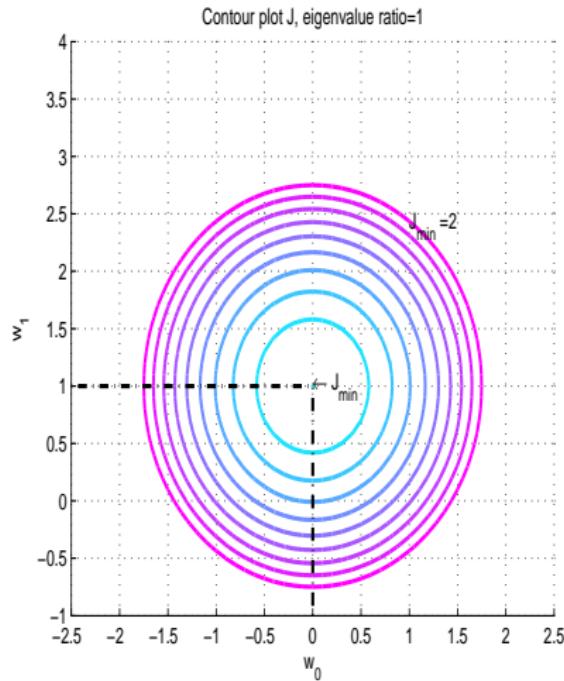
**General expression:**  $J = J_{min} + (\underline{w} - \underline{w}_o)^t \cdot \mathbf{R}_x \cdot (\underline{w} - \underline{w}_o)$

$$J_{min} = J_{\underline{w}=\underline{w}_o} = E\{r^2[k]\} = E\{r[k] \cdot e[k]\} = E\{e^2[k]\} - \underline{r}_{ex}^t \mathbf{R}_x^{-1} \underline{r}_{ex}$$

From general expression  $\Rightarrow J$  quadratic in  $\underline{w}$  thus  $\underline{w}_o$  really minimum

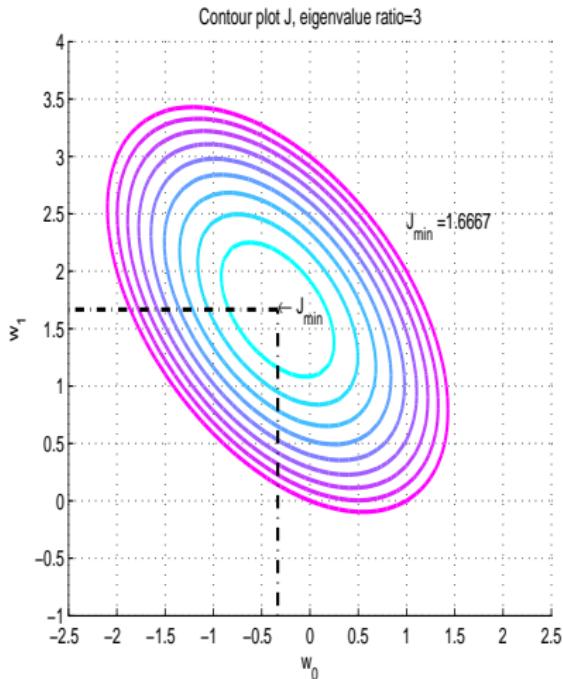
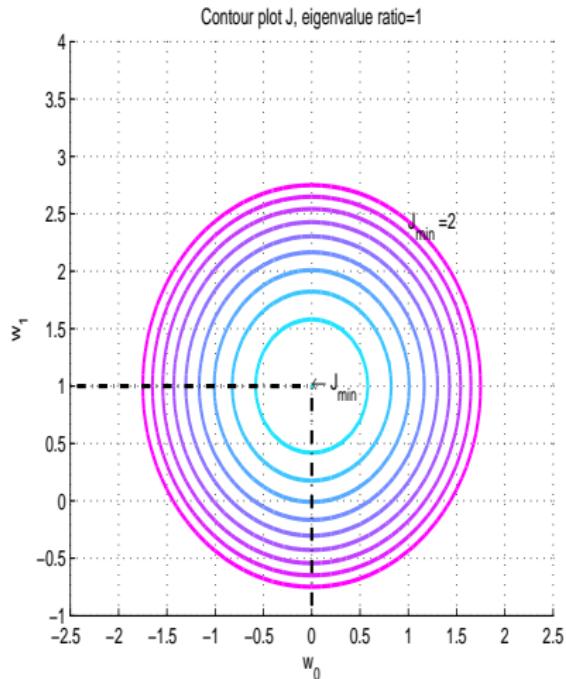
## Fixed weights: MMSE

Contour plots  $J = J_{min} + (\underline{w} - \underline{w}_o)^t \cdot R_x \cdot (\underline{w} - \underline{w}_o)$



# Fixed weights: MMSE

Contour plots  $J = J_{min} + (\underline{w} - \underline{w}_o)^t \cdot R_x \cdot (\underline{w} - \underline{w}_o)$



Eigenvalues: see Appendix

## Two MMSE variants

## Two MMSE variants

- **Complex MMSE:**

Setup with complex signals and weights

Similar result as before:

$$\underline{w}_o = \underline{R}_x^{-1} \cdot \underline{r}_{e^*x}$$

with  $\underline{r}_{e^*x} = E\{e^*[k]\underline{x}[k]\}$  and  $\underline{R}_x = E\{\underline{x}[k] \cdot \underline{x}^h[k]\}$  ( $h=\text{hermetian}$ )

## Two MMSE variants

- **Complex MMSE:**

Setup with complex signals and weights

Similar result as before:

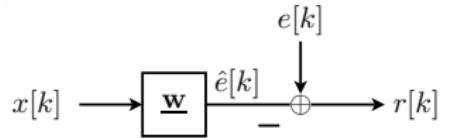
$$\underline{w}_o = \underline{R}_x^{-1} \cdot \underline{r}_{e^*x}$$

with  $\underline{r}_{e^*x} = E\{e^*[k]\underline{x}[k]\}$  and  $\underline{R}_x = E\{\underline{x}[k] \cdot \underline{x}^h[k]\}$  ( $h=\text{hermitian}$ )

- **Constrained MMSE:**

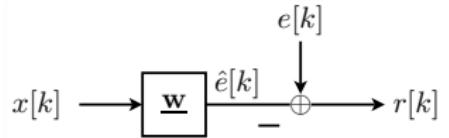
Setup with set of constraints on weights

# Constrained MMSE



**Goal:** Minimize  $E\{r^2[k]\}$  subject to  $M$  constraints

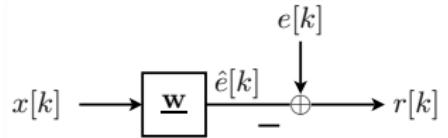
# Constrained MMSE



**Goal:** Minimize  $E\{r^2[k]\}$  subject to  $M$  constraints

Example: 
$$\sum_{i=0}^{N-1} w_i = 1$$

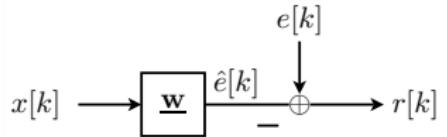
# Constrained MMSE



**Goal:** Minimize  $E\{r^2[k]\}$  subject to  $M$  constraints

Example:  $\sum_{i=0}^{N-1} w_i = 1 \quad \Leftrightarrow \quad (1, 1, \dots, 1)^t \cdot \begin{pmatrix} w_0 \\ w_1 \\ \vdots \\ w_{N-1} \end{pmatrix} = 1$

# Constrained MMSE



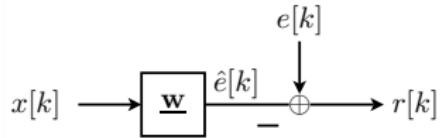
**Goal:** Minimize  $E\{r^2[k]\}$  subject to  $M$  constraints

Example:  $\sum_{i=0}^{N-1} w_i = 1 \quad \Leftrightarrow \quad (1, 1, \dots, 1)^t \cdot \begin{pmatrix} w_0 \\ w_1 \\ \vdots \\ w_{N-1} \end{pmatrix} = 1$

General:

$$\begin{aligned} \underline{\mathbf{c}}_1^t \cdot \underline{\mathbf{w}} &= f_1 \\ \underline{\mathbf{c}}_2^t \cdot \underline{\mathbf{w}} &= f_2 \\ &\vdots \\ \underline{\mathbf{c}}_M^t \cdot \underline{\mathbf{w}} &= f_M \end{aligned}$$

# Constrained MMSE



**Goal:** Minimize  $E\{r^2[k]\}$  subject to  $M$  constraints

Example:  $\sum_{i=0}^{N-1} w_i = 1 \Leftrightarrow (1, 1, \dots, 1)^t \cdot \begin{pmatrix} w_0 \\ w_1 \\ \vdots \\ w_{N-1} \end{pmatrix} = 1$

General:  $\begin{aligned} \underline{\mathbf{c}}_1^t \cdot \underline{\mathbf{w}} &= f_1 \\ \underline{\mathbf{c}}_2^t \cdot \underline{\mathbf{w}} &= f_2 \\ &\vdots \\ \underline{\mathbf{c}}_M^t \cdot \underline{\mathbf{w}} &= f_M \end{aligned} \Leftrightarrow \begin{pmatrix} \underline{\mathbf{c}}_1^t \\ \underline{\mathbf{c}}_2^t \\ \vdots \\ \underline{\mathbf{c}}_M^t \end{pmatrix} \cdot \underline{\mathbf{w}} = \underline{\mathbf{f}} \Leftrightarrow \underline{\mathbf{C}}^t \cdot \underline{\mathbf{w}} = \underline{\mathbf{f}}$

# Constrained MMSE

*Some notes on solving*     $C^t \cdot \underline{f} = \underline{w}$

## Constrained MMSE

*Some notes on solving  $C^t \cdot \underline{f} = \underline{w}$*

$M \times 1$  constraint vector :  $\underline{f} = (f_1, f_2, \dots, f_M)^t$

$N \times M$  constraint matrix :  $C = (\underline{c}_1, \underline{c}_2, \dots, \underline{c}_M)^t$

Note:  $M$  independent constraints  $\Rightarrow C$  full rank

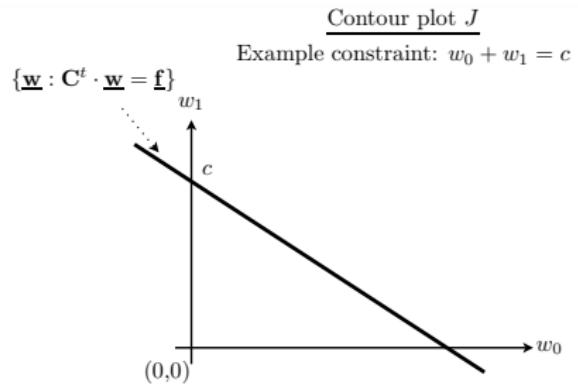
# Constrained MMSE

*Some notes on solving  $C^t \cdot \underline{f} = \underline{w}$*

$M \times 1$  constraint vector :  $\underline{f} = (f_1, f_2, \dots, f_M)^t$

$N \times M$  constraint matrix :  $C = (\underline{c}_1, \underline{c}_2, \dots, \underline{c}_M)^t$

Note:  $M$  independent constraints  $\Rightarrow C$  full rank



# Constrained MMSE

**Solutions of  $C^t \cdot \underline{w} = \underline{f}$**

# Constrained MMSE

**Solutions of  $C^t \cdot \underline{w} = \underline{f}$**

- ▶ Case  $N = M$ :

$$\Rightarrow \underline{w}^c = (C^t)^{-1} \cdot \underline{f}$$

$\Rightarrow$  No degrees of freedom left for MMSE

# Constrained MMSE

**Solutions of  $C^t \cdot \underline{w} = \underline{f}$**

► Case  $N = M$ :

⇒  $\underline{w}^c = (C^t)^{-1} \cdot \underline{f}$

⇒ *No degrees of freedom left for MMSE*

► Case  $N > M$ :

⇒ Possible solution  $\underline{w}^c = (C^t)^\dagger \cdot \underline{f}$

Appendix: Generalized inverse  $(C^t)^\dagger = C \cdot (C^t \cdot C)^{-1}$

⇒  *$N - M$  degrees of freedom left over for MMSE*

# Constrained MMSE

**Solutions of  $C^t \cdot \underline{w} = \underline{f}$**

► Case  $N = M$ :

⇒  $\underline{w}^c = (C^t)^{-1} \cdot \underline{f}$

⇒ *No degrees of freedom left for MMSE*

► Case  $N > M$ :

⇒ Possible solution  $\underline{w}^c = (C^t)^\dagger \cdot \underline{f}$

Appendix: Generalized inverse  $(C^t)^\dagger = C \cdot (C^t \cdot C)^{-1}$

⇒  $N - M$  degrees of freedom left over for MMSE

► Case  $N < M$ :

⇒ *Conflicting solutions*

⇒ Choose e.g. minimum norm solution

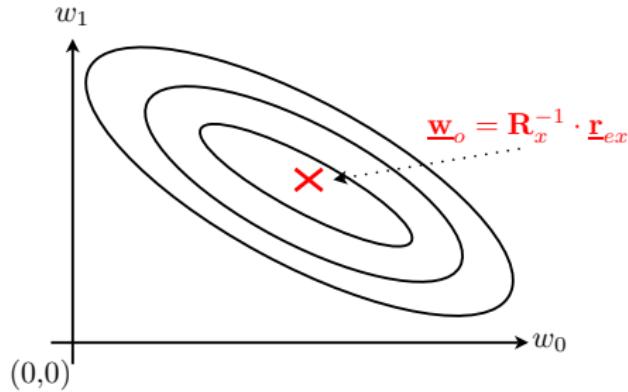
## Constrained MMSE

We can't reach  $\underline{w}_o$ , but we can do better than  $\underline{w}^c$ :

# Constrained MMSE

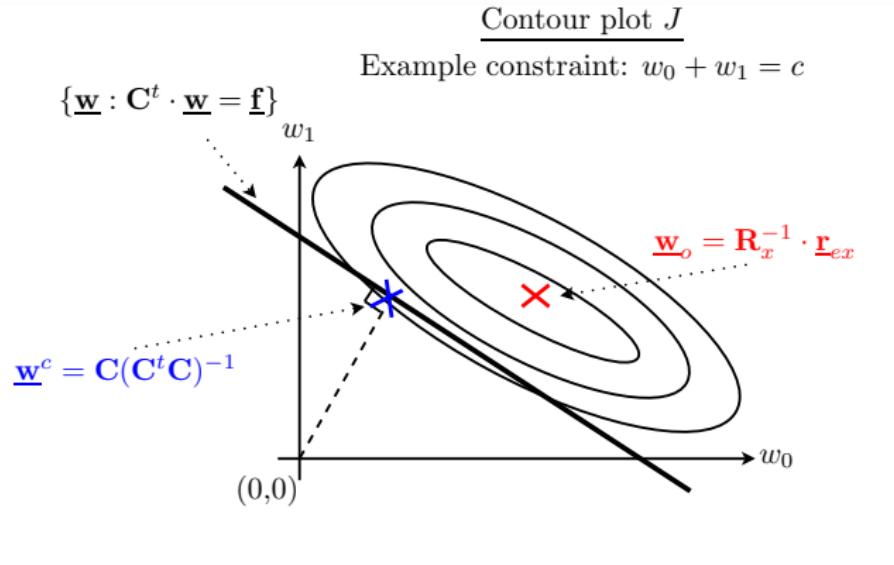
We can't reach  $\underline{w}_o$ , but we can do better than  $\underline{w}^c$ :

Contour plot  $J$



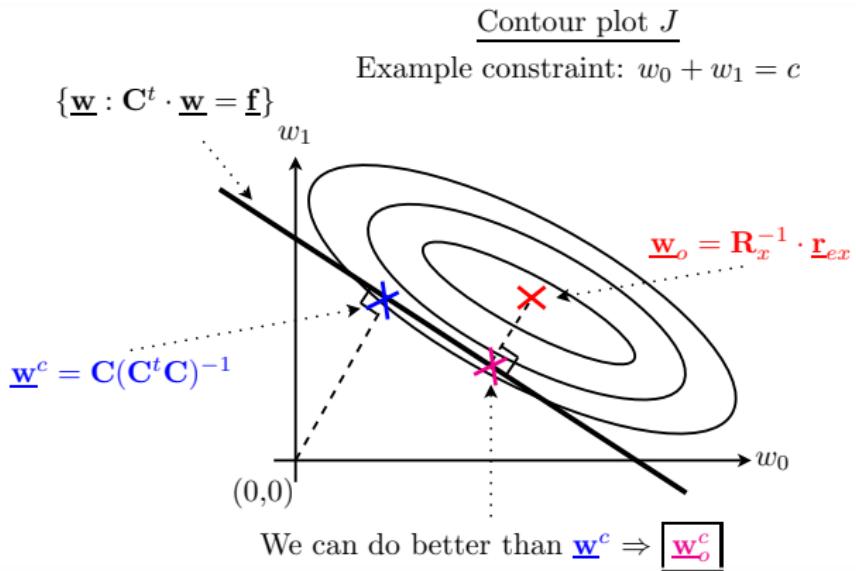
# Constrained MMSE

We can't reach  $\underline{w}_o$ , but we can do better than  $\underline{w}^c$ :



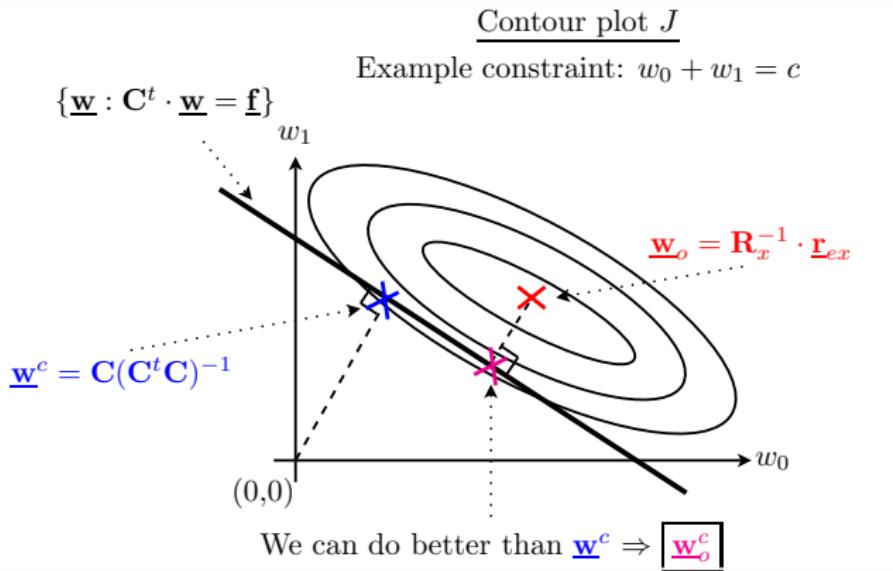
# Constrained MMSE

We can't reach  $\underline{w}_o$ , but we can do better than  $\underline{w}^c$ :



# Constrained MMSE

We can't reach  $\underline{w}_o$ , but we can do better than  $\underline{w}^c$ :



Use  $N - M$  degrees of freedom to improve result:  $\underline{w}^c \Rightarrow \underline{w}_o$

# Constrained MMSE

**Use Lagrange multipliers**

# Constrained MMSE

**Use Lagrange multipliers**

Performance index:

$$\begin{aligned} J^c &= E\{r^2\} + \underline{\lambda}^t (C^t \underline{w} - \underline{f}) \\ &= E\{e^2\} - \underline{w}^t \underline{r}_{ex} - \underline{r}_{ex}^t \underline{w} + \underline{w}^t R_x \underline{w} + \underline{\lambda}^t (C^t \underline{w} - \underline{f}) \end{aligned}$$

# Constrained MMSE

**Use Lagrange multipliers**

Performance index:

$$\begin{aligned} J^c &= E\{r^2\} + \underline{\lambda}^t(C^t \underline{w} - \underline{f}) \\ &= E\{e^2\} - \underline{w}^t \underline{r}_{ex} - \underline{r}_{ex}^t \underline{w} + \underline{w}^t R_x \underline{w} + \underline{\lambda}^t(C^t \underline{w} - \underline{f}) \end{aligned}$$

Gradient vector:

$$\frac{dJ^c}{d\underline{w}} = -2\underline{r}_{ex} + 2R_x \underline{w} + C \underline{\lambda}$$

$$\frac{dJ^c}{d\underline{w}} = \underline{0} \Rightarrow \underline{w}_o^c = R_x^{-1} \underline{r}_{ex} - \frac{1}{2} R_x^{-1} C \underline{\lambda}$$

# Constrained MMSE

**Use Lagrange multipliers**

Performance index:

$$\begin{aligned} J^c &= E\{r^2\} + \underline{\lambda}^t(C^t \underline{w} - \underline{f}) \\ &= E\{e^2\} - \underline{w}^t \underline{r}_{ex} - \underline{r}_{ex}^t \underline{w} + \underline{w}^t R_x \underline{w} + \underline{\lambda}^t(C^t \underline{w} - \underline{f}) \end{aligned}$$

Gradient vector:

$$\frac{dJ^c}{d\underline{w}} = -2\underline{r}_{ex} + 2R_x \underline{w} + C\underline{\lambda}$$

$$\frac{dJ^c}{d\underline{w}} = \underline{0} \Rightarrow \underline{w}_o^c = R_x^{-1} \underline{r}_{ex} - \frac{1}{2} R_x^{-1} C \underline{\lambda}$$

Furthermore in optimum:  $C^t \underline{w}_o^c = \underline{f}$

# Constrained MMSE

**Use Lagrange multipliers**

Performance index:

$$\begin{aligned} J^c &= E\{r^2\} + \underline{\lambda}^t(C^t \underline{w} - \underline{f}) \\ &= E\{e^2\} - \underline{w}^t \underline{r}_{ex} - \underline{r}_{ex}^t \underline{w} + \underline{w}^t R_x \underline{w} + \underline{\lambda}^t(C^t \underline{w} - \underline{f}) \end{aligned}$$

Gradient vector:

$$\frac{dJ^c}{d\underline{w}} = -2\underline{r}_{ex} + 2R_x \underline{w} + C \underline{\lambda}$$

$$\frac{dJ^c}{d\underline{w}} = \underline{0} \Rightarrow \underline{w}_o^c = R_x^{-1} \underline{r}_{ex} - \frac{1}{2} R_x^{-1} C \underline{\lambda}$$

Furthermore in optimum:  $C^t \underline{w}_o^c = \underline{f}$

Combine last two equations:

$$\Rightarrow \underline{\lambda} = 2(C^t R_x^{-1} C)^{-1}(C^t R_x^{-1} \underline{r}_{ex} - \underline{f})$$

## Constrained MMSE

$$\Rightarrow \underline{w}_o^c = \underline{w}_o + R_x^{-1}C(C^t R_x^{-1} C)^{-1}(\underline{f} - C^t \underline{w}_o)$$

with

$$\underline{w}_o = R_x^{-1} \underline{r}_{ex}$$

## Constrained MMSE

$$\Rightarrow \underline{w}_o^c = \underline{w}_o + R_x^{-1}C(C^t R_x^{-1} C)^{-1}(\underline{f} - C^t \underline{w}_o)$$

with

$$\underline{w}_o = R_x^{-1} \underline{r}_{ex}$$

Similar result:

$$\underline{w}_o^c = R_x^{-1}C(C^t R_x^{-1} C)^{-1}\underline{f}$$

## Constrained MMSE

$$\Rightarrow \underline{w}_o^c = \underline{w}_o + R_x^{-1}C(C^t R_x^{-1} C)^{-1}(\underline{f} - C^t \underline{w}_o)$$

with

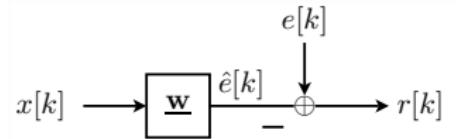
$$\underline{w}_o = R_x^{-1} \underline{r}_{ex}$$

Similar result:

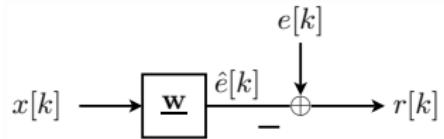
$$\underline{w}_o^c = R_x^{-1}C(C^t R_x^{-1} C)^{-1}\underline{f}$$

Check:  $C^t \underline{w}_o^c = C^t \underline{w}_o + (C^t R_x^{-1} C)(C^t R_x^{-1} C)^{-1}(\underline{f} - C^t \underline{w}_o) = \underline{f}$

# Least Squares (LS)



# Least Squares (LS)



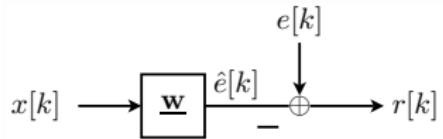
Different quadratic cost functions:

- ▶ Mean Square Error (MSE):

$$J_{mse} = E\{r^2[k]\} = E\{(e[k] - \underline{\mathbf{w}}^t \underline{x}[k])^2\}$$

$\Rightarrow$  Minimum MSE (MMSE) = Wiener

# Least Squares (LS)



Different quadratic cost functions:

- ▶ Mean Square Error (MSE):

$$J_{mse} = E\{r^2[k]\} = E\{(e[k] - \underline{\mathbf{w}}^T \underline{x}[k])^2\}$$

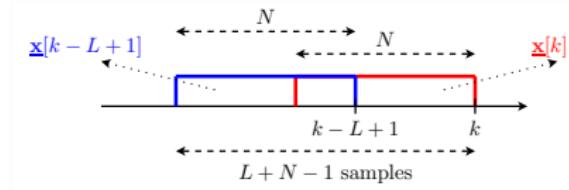
$\Rightarrow$  Minimum MSE (MMSE) = Wiener

- ▶ **Least Square (LS):** If statistical information is not available

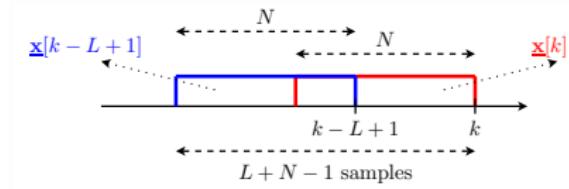
$\Rightarrow$

Use criterion based on data (thus without  $E\{\cdot\}$ )

Collect  $L$  ( $\geq 1$ ) data vectors  $\underline{x}[k - i]$  (each of length  $N$ )

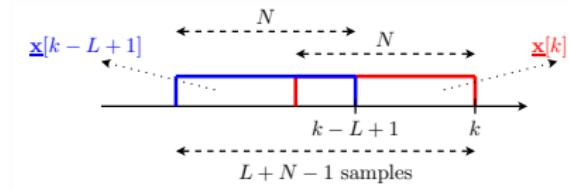


Collect  $L$  ( $\geq 1$ ) data vectors  $\underline{x}[k - i]$  (each of length  $N$ )



Available data (for  $i = 0, 1, \dots, L-1$ ):

Collect  $L$  ( $\geq 1$ ) data vectors  $\underline{x}[k - i]$  (each of length  $N$ )



Available data (for  $i = 0, 1, \dots, L - 1$ ):

- *Input signal samples/ vectors  $\underline{x}[k - i]$*
- *Reference signal samples:  $e[k - i]$*
- *Residual signal samples:  $r[k - i] = e[k - i] - \underline{x}^t[k - i] \cdot \underline{w}$*

Notation:

$$\underline{X}[k] = \begin{pmatrix} \underline{x}^t[k] \\ \underline{x}^t[k-1] \\ \vdots \\ \underline{x}^t[k-L+1] \end{pmatrix}$$

$$\underline{e}[k] = \begin{pmatrix} e[k] \\ e[k-1] \\ \vdots \\ e[k-L+1] \end{pmatrix}$$

$$\underline{w} = \begin{pmatrix} w_0 \\ w_1 \\ \vdots \\ w_{N-1} \end{pmatrix}$$

$$\underline{r}[k] = \begin{pmatrix} r[k] \\ r[k-1] \\ \vdots \\ r[k-L+1] \end{pmatrix}$$

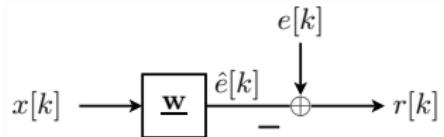
Notation:

$$X[k] = \begin{pmatrix} \underline{x}^t[k] \\ \underline{x}^t[k-1] \\ \vdots \\ \underline{x}^t[k-L+1] \end{pmatrix} \quad \underline{w} = \begin{pmatrix} w_0 \\ w_1 \\ \vdots \\ w_{N-1} \end{pmatrix}$$

$$\underline{e}[k] = \begin{pmatrix} e[k] \\ e[k-1] \\ \vdots \\ e[k-L+1] \end{pmatrix} \quad \underline{r}[k] = \begin{pmatrix} r[k] \\ r[k-1] \\ \vdots \\ r[k-L+1] \end{pmatrix}$$

Simplified notation (skip time indices):

$$\underline{r} = \underline{e} - X \cdot \underline{w}$$



**LS problem formulation:**

$$\underline{\mathbf{w}}_{ls,o} = \arg \min_{\underline{\mathbf{w}}} |\underline{\mathbf{e}} - \mathbf{X} \cdot \underline{\mathbf{w}}|^2$$

$$J_{ls} = \sum_{i=0}^{L-1} r^2[k-i] = \underline{r}^t \cdot \underline{r} = (\underline{e}^t - \underline{w}^t \mathbf{X}^t) \cdot (\underline{e} - \mathbf{X} \underline{w})$$

$$\begin{aligned} J_{ls} &= \sum_{i=0}^{L-1} r^2[k-i] = \underline{r}^t \cdot \underline{r} = (\underline{\mathbf{e}}^t - \underline{\mathbf{w}}^t \mathbf{X}^t) \cdot (\underline{\mathbf{e}} - \mathbf{X} \underline{\mathbf{w}}) \\ &= \underline{\mathbf{e}}^t \underline{\mathbf{e}} + \underline{\mathbf{w}}^t \mathbf{X}^t \mathbf{X} \underline{\mathbf{w}} - \underline{\mathbf{w}}^t \mathbf{X}^t \underline{\mathbf{e}} - \underline{\mathbf{e}}^t \mathbf{X} \underline{\mathbf{w}} \end{aligned}$$

$$\begin{aligned} J_{ls} &= \sum_{i=0}^{L-1} r^2[k-i] = \underline{r}^t \cdot \underline{r} = (\underline{\mathbf{e}}^t - \underline{\mathbf{w}}^t \mathbf{X}^t) \cdot (\underline{\mathbf{e}} - \mathbf{X} \underline{\mathbf{w}}) \\ &= \underline{\mathbf{e}}^t \underline{\mathbf{e}} + \underline{\mathbf{w}}^t \mathbf{X}^t \mathbf{X} \underline{\mathbf{w}} - \underline{\mathbf{w}}^t \mathbf{X}^t \underline{\mathbf{e}} - \underline{\mathbf{e}}^t \mathbf{X} \underline{\mathbf{w}} \end{aligned}$$

Minimum by setting gradient equal to zero:

$$\frac{dJ_{ls}}{d\underline{\mathbf{w}}} = \underline{\bigtriangledown}_{ls} = -2(\mathbf{X}^t \underline{\mathbf{e}} - \mathbf{X}^t \mathbf{X} \cdot \underline{\mathbf{w}}) = 0$$

$$\begin{aligned} J_{ls} &= \sum_{i=0}^{L-1} r^2[k-i] = \underline{r}^t \cdot \underline{r} = (\underline{\mathbf{e}}^t - \underline{\mathbf{w}}^t \mathbf{X}^t) \cdot (\underline{\mathbf{e}} - \mathbf{X} \underline{\mathbf{w}}) \\ &= \underline{\mathbf{e}}^t \underline{\mathbf{e}} + \underline{\mathbf{w}}^t \mathbf{X}^t \mathbf{X} \underline{\mathbf{w}} - \underline{\mathbf{w}}^t \mathbf{X}^t \underline{\mathbf{e}} - \underline{\mathbf{e}}^t \mathbf{X} \underline{\mathbf{w}} \end{aligned}$$

Minimum by setting gradient equal to zero:

$$\frac{dJ_{ls}}{d\underline{\mathbf{w}}} = \underline{\nabla}_{ls} = -2(\mathbf{X}^t \underline{\mathbf{e}} - \mathbf{X}^t \mathbf{X} \cdot \underline{\mathbf{w}}) = \underline{0}$$

With  $\bar{\mathbf{R}} = \mathbf{X}^t \mathbf{X}$  and  $\bar{\mathbf{r}} = \mathbf{X}^t \underline{\mathbf{e}}$

$$\begin{aligned}
 J_{ls} &= \sum_{i=0}^{L-1} r^2[k-i] = \underline{r}^t \cdot \underline{r} = (\underline{\mathbf{e}}^t - \underline{\mathbf{w}}^t \mathbf{X}^t) \cdot (\underline{\mathbf{e}} - \mathbf{X} \underline{\mathbf{w}}) \\
 &= \underline{\mathbf{e}}^t \underline{\mathbf{e}} + \underline{\mathbf{w}}^t \mathbf{X}^t \mathbf{X} \underline{\mathbf{w}} - \underline{\mathbf{w}}^t \mathbf{X}^t \underline{\mathbf{e}} - \underline{\mathbf{e}}^t \mathbf{X} \underline{\mathbf{w}}
 \end{aligned}$$

Minimum by setting gradient equal to zero:

$$\frac{dJ_{ls}}{d\underline{\mathbf{w}}} = \underline{\bigtriangledown}_{ls} = -2(\mathbf{X}^t \underline{\mathbf{e}} - \mathbf{X}^t \mathbf{X} \cdot \underline{\mathbf{w}}) = \underline{0}$$

With  $\bar{\mathbf{R}} = \mathbf{X}^t \mathbf{X}$  and  $\bar{\mathbf{r}} = \mathbf{X}^t \underline{\mathbf{e}}$

$\Rightarrow$  **Normal Equations**

$$\bar{\mathbf{R}}_x \cdot \underline{\mathbf{w}} = \bar{\mathbf{r}}_{ex}$$

$\Rightarrow$  **Wiener filter**

$$\underline{\mathbf{w}}_{ls,o} = \bar{\mathbf{R}}_x^{-1} \cdot \bar{\mathbf{r}}_{ex}$$

## LS: Correspondence with MMSE

Use time-averaging (ergodicity):

$$\hat{\underline{R}}_x = \frac{1}{L} \sum_{i=0}^{L-1} \underline{x}[k-i] \cdot \underline{x}^t[k-i] = \frac{1}{L} \underline{X}^t \cdot \underline{X} = \frac{1}{L} \bar{\underline{R}}_x$$

$$\hat{\underline{r}}_{\text{ex}} = \frac{1}{L} \sum_{i=0}^{L-1} \underline{x}[k-i] \cdot \underline{e}[k-i] = \frac{1}{L} \underline{X}^t \cdot \underline{e} = \frac{1}{L} \bar{\underline{r}}_{\text{ex}}$$

## LS: Correspondence with MMSE

Use time-averaging (ergodicity):

$$\hat{R}_x = \frac{1}{L} \sum_{i=0}^{L-1} \underline{x}[k-i] \cdot \underline{x}^t[k-i] = \frac{1}{L} \underline{X}^t \cdot \underline{X} = \frac{1}{L} \bar{R}_x$$

$$\hat{r}_{ex} = \frac{1}{L} \sum_{i=0}^{L-1} \underline{x}[k-i] \cdot e[k-i] = \frac{1}{L} \underline{X}^t \cdot \underline{e} = \frac{1}{L} \bar{r}_{ex}$$

with  $\hat{R}_x$  estimate of  $R_x$  and  $\hat{r}_{ex}$  estimate of  $r_{ex}$

$$\Rightarrow \hat{w}_{mmse} = \left( \frac{1}{L} \bar{R}_x \right)^{-1} \cdot \left( \frac{1}{L} \bar{r}_{ex} \right) = \bar{R}_x^{-1} \cdot \bar{r}_{ex} = \underline{w}_{ls}$$

## LS: Correspondence with MMSE

Use time-averaging (ergodicity):

$$\hat{R}_x = \frac{1}{L} \sum_{i=0}^{L-1} \underline{x}[k-i] \cdot \underline{x}^t[k-i] = \frac{1}{L} \underline{X}^t \cdot \underline{X} = \frac{1}{L} \bar{R}_x$$

$$\hat{r}_{ex} = \frac{1}{L} \sum_{i=0}^{L-1} \underline{x}[k-i] \cdot \underline{e}[k-i] = \frac{1}{L} \underline{X}^t \cdot \underline{e} = \frac{1}{L} \bar{r}_{ex}$$

with  $\hat{R}_x$  estimate of  $R_x$  and  $\hat{r}_{ex}$  estimate of  $r_{ex}$

$$\Rightarrow \hat{\underline{w}}_{mmse} = \left( \frac{1}{L} \bar{R}_x \right)^{-1} \cdot \left( \frac{1}{L} \bar{r}_{ex} \right) = \bar{R}_x^{-1} \cdot \bar{r}_{ex} = \underline{w}_{ls}$$

Finally note that for ergodic processes:

$$\lim_{L \rightarrow \infty} \frac{1}{L} \bar{R}_x = R_x ; \quad \lim_{L \rightarrow \infty} \frac{1}{L} \bar{r}_{ex} = r_{ex} ; \quad \lim_{L \rightarrow \infty} \underline{w}_{ls} = \underline{w}_{mmse}$$

## Steepest gradient descent (SGD)

**Problem:** Optimal Wiener involves  $R_x^{-1}$

## Steepest gradient descent (SGD)

**Problem:** Optimal Wiener involves  $R_x^{-1}$

To avoid this inversion, estimate optimum *iteratively*

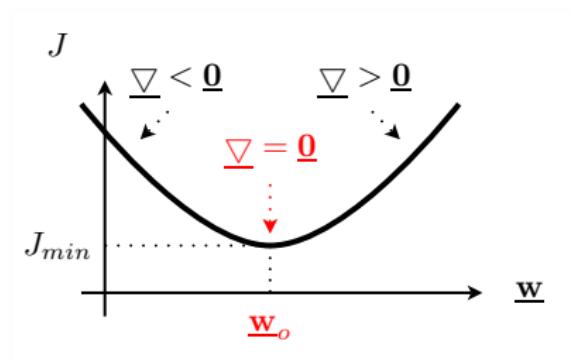
# Steepest gradient descent (SGD)

**Problem:** Optimal Wiener involves  $R_x^{-1}$

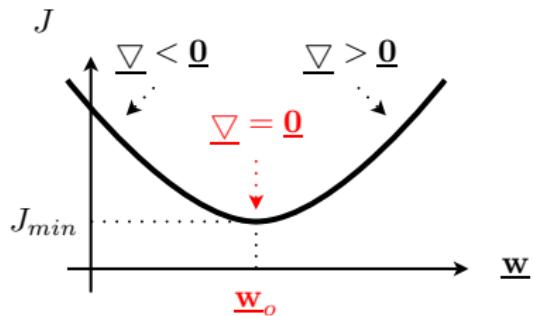
To avoid this inversion, estimate optimum *iteratively*

**Goal:** Decrease  $J$  each new iteration

# SGD



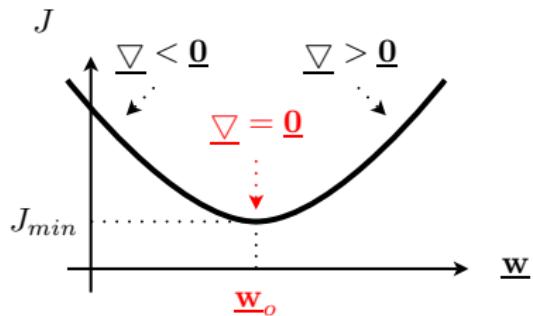
# SGD



**SGD principle:** Update in negative gradient direction

$$\Leftrightarrow \underline{w} \doteq \underline{w} - \alpha \underline{\nabla} \text{ with adaptation constant } \alpha \geq 0$$

# SGD

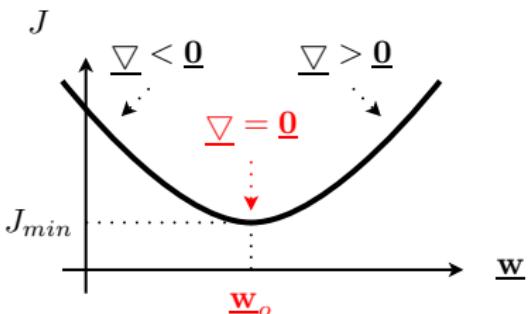


**SGD principle:** Update in negative gradient direction

$$\Leftrightarrow \dot{w} = w - \alpha \nabla \text{ with adaptation constant } \alpha \geq 0$$

With  $\nabla = -2(r_{ex} - R_x w[k])$

# SGD



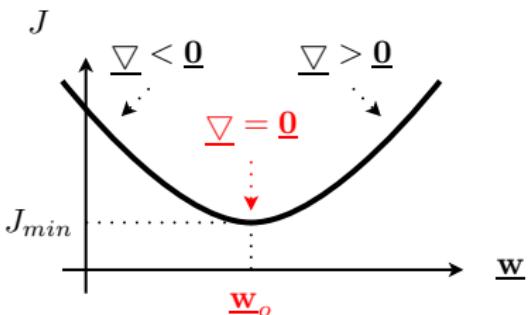
**SGD principle:** Update in negative gradient direction

$$\Leftrightarrow \underline{w} \doteq \underline{w} - \alpha \underline{\nabla} \text{ with adaptation constant } \alpha \geq 0$$

With  $\underline{\nabla} = -2(r_{ex} - R_x \underline{w}[k]) \Rightarrow \text{SGD algorithm:}$

$$\underline{w}[k+1] = \underline{w}[k] + 2\alpha(r_{ex} - R_x \underline{w}[k])$$

# SGD



**SGD principle:** Update in negative gradient direction

$$\Leftrightarrow \underline{w} \doteq \underline{w} - \alpha \underline{\nabla} \text{ with adaptation constant } \alpha \geq 0$$

With  $\underline{\nabla} = -2(r_{ex} - R_x \underline{w}[k]) \Rightarrow \text{SGD algorithm:}$

$$\boxed{\underline{w}[k+1] = \underline{w}[k] + 2\alpha(r_{ex} - R_x \underline{w}[k])}$$

Notes: 1) No matrix inversion needed! 2) Usually  $\underline{w}[0] = 0$

# SGD

SGD converges to Wiener solution:

$$\lim_{k \rightarrow \infty} \underline{w}[k] \simeq R_x^{-1} \cdot r_{ex}$$

# SGD

SGD converges to Wiener solution:

$$\lim_{k \rightarrow \infty} \underline{w}[k] \simeq R_x^{-1} \cdot r_{ex}$$

'Proof':

For  $k \rightarrow \infty$  we have:

$$\underline{w}[k+1] \simeq \underline{w}[k] \simeq \underline{w}[\infty]$$

# SGD

SGD converges to Wiener solution:

$$\lim_{k \rightarrow \infty} \underline{w}[k] \simeq R_x^{-1} \cdot r_{ex}$$

'Proof':

For  $k \rightarrow \infty$  we have:

$$\underline{w}[k+1] \simeq \underline{w}[k] \simeq \underline{w}[\infty]$$

$$\text{SGD} \Rightarrow \underline{w}[\infty] \simeq \underline{w}[\infty] + 2\alpha(r_{ex} - R_x \underline{w}[\infty])$$

# SGD

SGD converges to Wiener solution:

$$\lim_{k \rightarrow \infty} \underline{w}[k] \simeq R_x^{-1} \cdot r_{ex}$$

'Proof':

For  $k \rightarrow \infty$  we have:

$$\underline{w}[k+1] \simeq \underline{w}[k] \simeq \underline{w}[\infty]$$

$$\begin{aligned} \text{SGD} \Rightarrow \underline{w}[\infty] &\simeq \underline{w}[\infty] + 2\alpha(r_{ex} - R_x \underline{w}[\infty]) \\ \Rightarrow \underline{w}[\infty] &\simeq R_x^{-1} \cdot r_{ex} \end{aligned}$$

# SGD

SGD converges to Wiener solution:

$$\lim_{k \rightarrow \infty} \underline{w}[k] \simeq R_x^{-1} \cdot r_{ex}$$

'Proof':

For  $k \rightarrow \infty$  we have:

$$\underline{w}[k+1] \simeq \underline{w}[k] \simeq \underline{w}[\infty]$$

$$\begin{aligned} \text{SGD} \Rightarrow \underline{w}[\infty] &\simeq \underline{w}[\infty] + 2\alpha(r_{ex} - R_x \underline{w}[\infty]) \\ \Rightarrow \underline{w}[\infty] &\simeq R_x^{-1} \cdot r_{ex} \end{aligned}$$

For exact proof we need **stability analysis**

# Stability SGD

Define difference weight vector:  $\underline{d}[k] = \underline{w}[k] - \underline{w}_o$

## Stability SGD

Define difference weight vector:  $\underline{d}[k] = \underline{w}[k] - \underline{w}_o$

$$\underline{w}[k + 1] = \underline{w}[k] + 2\alpha(\underline{r}_{ex} - R_x \underline{w}[k])$$

# Stability SGD

Define difference weight vector:  $\underline{d}[k] = \underline{w}[k] - \underline{w}_o$

$$\underline{w}[k + 1] = \underline{w}[k] + 2\alpha(\underline{r}_{ex} - R_x \underline{w}[k])$$

$$\underline{w}[k + 1] - \underline{w}_o = (I - 2\alpha R_x) \cdot \underline{w}[k] - \underline{w}_o + 2\alpha \underline{r}_{ex}$$

# Stability SGD

Define difference weight vector:  $\underline{d}[k] = \underline{w}[k] - \underline{w}_o$

$$\begin{aligned}\underline{w}[k+1] &= \underline{w}[k] + 2\alpha(\underline{r}_{ex} - R_x \underline{w}[k]) \\ \underline{w}[k+1] - \underline{w}_o &= (I - 2\alpha R_x) \cdot \underline{w}[k] - \underline{w}_o + 2\alpha \underline{r}_{ex} \\ \Rightarrow \underline{d}[k+1] &= (I - 2\alpha R_x) \cdot \underline{d}[k]\end{aligned}$$

# Stability SGD

Define difference weight vector:  $\underline{d}[k] = \underline{w}[k] - \underline{w}_o$

$$\begin{aligned}\underline{w}[k+1] &= \underline{w}[k] + 2\alpha(\underline{r}_{ex} - R_x \underline{w}[k]) \\ \underline{w}[k+1] - \underline{w}_o &= (I - 2\alpha R_x) \cdot \underline{w}[k] - \underline{w}_o + 2\alpha \underline{r}_{ex} \\ \Rightarrow \underline{d}[k+1] &= (I - 2\alpha R_x) \cdot \underline{d}[k]\end{aligned}$$

Recursion:

$$\underline{d}[k] = (I - 2\alpha R_x) \cdot \underline{d}[k-1] = \dots = (I - 2\alpha R_x)^k \cdot \underline{d}[0]$$

# Stability SGD

Define difference weight vector:  $\underline{d}[k] = \underline{w}[k] - \underline{w}_o$

$$\begin{aligned}\underline{w}[k+1] &= \underline{w}[k] + 2\alpha(\underline{r}_{ex} - R_x \underline{w}[k]) \\ \underline{w}[k+1] - \underline{w}_o &= (I - 2\alpha R_x) \cdot \underline{w}[k] - \underline{w}_o + 2\alpha \underline{r}_{ex} \\ \Rightarrow \underline{d}[k+1] &= (I - 2\alpha R_x) \cdot \underline{d}[k]\end{aligned}$$

Recursion:

$$\underline{d}[k] = (I - 2\alpha R_x) \cdot \underline{d}[k-1] = \dots = (I - 2\alpha R_x)^k \cdot \underline{d}[0]$$

Stable iff:  $\lim_{k \rightarrow \infty} (I - 2\alpha R_x)^k = 0$

# Stability SGD

Define difference weight vector:  $\underline{d}[k] = \underline{w}[k] - \underline{w}_o$

$$\begin{aligned}\underline{w}[k+1] &= \underline{w}[k] + 2\alpha(\underline{r}_{ex} - R_x \underline{w}[k]) \\ \underline{w}[k+1] - \underline{w}_o &= (I - 2\alpha R_x) \cdot \underline{w}[k] - \underline{w}_o + 2\alpha \underline{r}_{ex} \\ \Rightarrow \underline{d}[k+1] &= (I - 2\alpha R_x) \cdot \underline{d}[k]\end{aligned}$$

Recursion:

$$\underline{d}[k] = (I - 2\alpha R_x) \cdot \underline{d}[k-1] = \dots = (I - 2\alpha R_x)^k \cdot \underline{d}[0]$$

Stable iff:  $\lim_{k \rightarrow \infty} (I - 2\alpha R_x)^k = 0$

Note:

When stable  $\Rightarrow \underline{d}[\infty] = \underline{0} \Rightarrow \underline{w}[\infty] \simeq \text{Wiener}$

# Stability SGD

How do weights converge:

# Stability SGD

## How do weights converge:

Use eigenvalue decomposition (see Appendix):

# Stability SGD

## How do weights converge:

Use eigenvalue decomposition (see Appendix):

With  $Q^h \cdot Q = Q \cdot Q^h = I$  and  $R_x = Q \Lambda Q^h$

# Stability SGD

## How do weights converge:

Use eigenvalue decomposition (see Appendix):

With  $Q^h \cdot Q = Q \cdot Q^h = I$  and  $R_x = Q \Lambda Q^h$

$$\begin{aligned}\Rightarrow (I - 2\alpha R_x)^k &= (QQ^h - 2\alpha Q\Lambda Q^h)^k \\ &= Q(I - 2\alpha\Lambda)^k Q^h\end{aligned}$$

# Stability SGD

## How do weights converge:

Use eigenvalue decomposition (see Appendix):

With  $Q^h \cdot Q = Q \cdot Q^h = I$  and  $R_x = Q \Lambda Q^h$

$$\begin{aligned}\Rightarrow (I - 2\alpha R_x)^k &= (QQ^h - 2\alpha Q\Lambda Q^h)^k \\ &= Q(I - 2\alpha\Lambda)^k Q^h\end{aligned}$$

Change of variables:  $\underline{D}[k] = Q^h \cdot \underline{d}[k]$

# Stability SGD

## How do weights converge:

Use eigenvalue decomposition (see Appendix):

With  $Q^h \cdot Q = Q \cdot Q^h = I$  and  $R_x = Q \Lambda Q^h$

$$\begin{aligned}\Rightarrow (I - 2\alpha R_x)^k &= (QQ^h - 2\alpha Q \Lambda Q^h)^k \\ &= Q(I - 2\alpha \Lambda)^k Q^h\end{aligned}$$

Change of variables:  $\underline{D}[k] = Q^h \cdot \underline{d}[k]$

$$\underline{d}[k] = (I - 2\alpha R_x)^k \underline{d}[0] \Rightarrow \underline{D}[k] = (I - 2\alpha \Lambda)^k \underline{D}[0]$$

# Stability SGD

## How do weights converge:

Use eigenvalue decomposition (see Appendix):

With  $Q^h \cdot Q = Q \cdot Q^h = I$  and  $R_x = Q \Lambda Q^h$

$$\begin{aligned}\Rightarrow (I - 2\alpha R_x)^k &= (QQ^h - 2\alpha Q \Lambda Q^h)^k \\ &= Q(I - 2\alpha \Lambda)^k Q^h\end{aligned}$$

Change of variables:  $\underline{D}[k] = Q^h \cdot \underline{d}[k]$

$$\underline{d}[k] = (I - 2\alpha R_x)^k \underline{d}[0] \Rightarrow \underline{D}[k] = (I - 2\alpha \Lambda)^k \underline{D}[0]$$

Recursion stable iff:  $\lim_{k \rightarrow \infty} (I - 2\alpha \Lambda)^k = 0$

# Stability SGD

Recursion stable iff:  $\lim_{k \rightarrow \infty} (\mathbf{I} - 2\alpha\Lambda)^k = 0$

# Stability SGD

Recursion stable iff:  $\lim_{k \rightarrow \infty} (\mathbf{I} - 2\alpha\Lambda)^k = 0$

Both matrices  $\mathbf{I}$  and  $\Lambda$  diagonal

## Stability SGD

Recursion stable iff:  $\lim_{k \rightarrow \infty} (\mathbf{I} - 2\alpha\Lambda)^k = \mathbf{0}$

Both matrices  $\mathbf{I}$  and  $\Lambda$  diagonal  $\Rightarrow$  Stable iff:

$$|1 - 2\alpha\lambda_i| < 1 \Leftrightarrow 0 < \alpha < \frac{1}{\lambda_i} \text{ for } i = 0, 1, \dots, N-1$$

# Stability SGD

Recursion stable iff:  $\lim_{k \rightarrow \infty} (\mathbf{I} - 2\alpha\Lambda)^k = \mathbf{0}$

Both matrices  $\mathbf{I}$  and  $\Lambda$  diagonal  $\Rightarrow$  Stable iff:

$$|1 - 2\alpha\lambda_i| < 1 \Leftrightarrow 0 < \alpha < \frac{1}{\lambda_i} \text{ for } i = 0, 1, \dots, N-1$$

Thus SGD algorithm stable iff:

$$0 < \alpha < \frac{1}{\lambda_{max}}$$

## Stability SGD

Recursion stable iff:  $\lim_{k \rightarrow \infty} (\mathbf{I} - 2\alpha\Lambda)^k = 0$

Both matrices  $\mathbf{I}$  and  $\Lambda$  diagonal  $\Rightarrow$  Stable iff:

$$|1 - 2\alpha\lambda_i| < 1 \Leftrightarrow 0 < \alpha < \frac{1}{\lambda_i} \text{ for } i = 0, 1, \dots, N-1$$

Thus SGD algorithm stable iff:

$$0 < \alpha < \frac{1}{\lambda_{max}}$$

For adaptation constant  $\alpha$  in this region:

$$\lim_{k \rightarrow \infty} \underline{\mathbf{w}}[k] = \underline{\mathbf{w}}_o = \mathbf{R}_x^{-1} \cdot \underline{\mathbf{r}}_{ex}$$

# Stability SGD

Recursion stable iff:  $\lim_{k \rightarrow \infty} (\mathbf{I} - 2\alpha\Lambda)^k = \mathbf{0}$

Both matrices  $\mathbf{I}$  and  $\Lambda$  diagonal  $\Rightarrow$  Stable iff:

$$|1 - 2\alpha\lambda_i| < 1 \Leftrightarrow 0 < \alpha < \frac{1}{\lambda_i} \text{ for } i = 0, 1, \dots, N-1$$

Thus SGD algorithm stable iff:

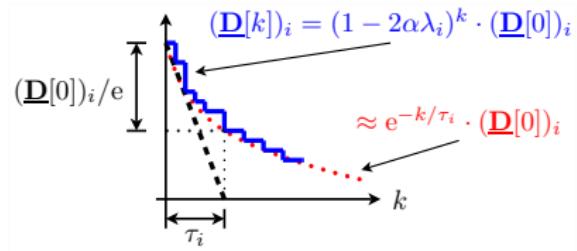
$$0 < \alpha < \frac{1}{\lambda_{\max}}$$

For adaptation constant  $\alpha$  in this region:

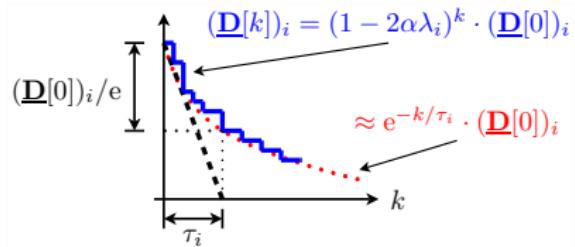
$$\lim_{k \rightarrow \infty} \underline{\mathbf{w}}[k] = \underline{\mathbf{w}}_o = \mathbf{R}_x^{-1} \cdot \underline{\mathbf{r}}_{ex}$$

$$J_{\underline{\mathbf{w}}=\underline{\mathbf{w}}_o} = E\{r^2[k]\} = J_{min} = E\{e^2\} - \underline{\mathbf{r}}_{ex}^t \mathbf{R}_x^{-1} \underline{\mathbf{r}}_{ex}$$

# Convergence SGD

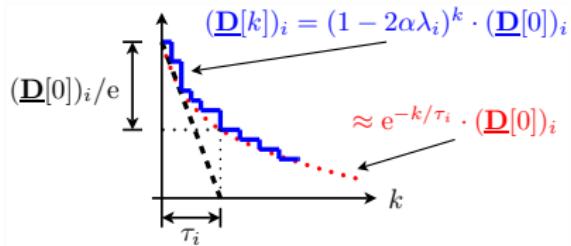


# Convergence SGD



$$e^{-k/\tau_i} \cdot (\underline{D}[0])_i \approx (1 - 2\alpha\lambda_i)^k \cdot (\underline{D}[0])_i \Rightarrow$$

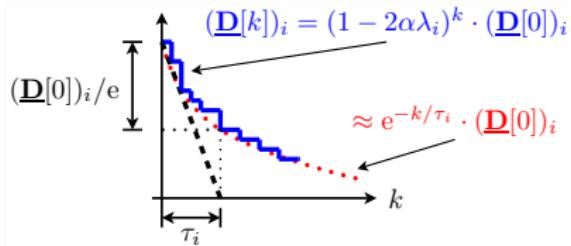
# Convergence SGD



$$e^{-k/\tau_i} \cdot (D[0])_i \approx (1 - 2\alpha\lambda_i)^k \cdot (D[0])_i \Rightarrow$$

Average behavior:  $\tau_{av,i} = \frac{-1}{\ln(1 - 2\alpha\lambda_i)}$

# Convergence SGD

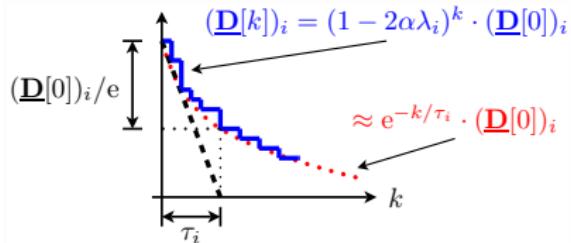


$$e^{-k/\tau_i} \cdot (\underline{D}[0])_i \approx (1 - 2\alpha\lambda_i)^k \cdot (\underline{D}[0])_i \Rightarrow$$

Average behavior:  $\tau_{av,i} = \frac{-1}{\ln(1 - 2\alpha\lambda_i)}$   $\Rightarrow$  For small  $\alpha$

$$\tau_{av,i} \approx \frac{1}{2\alpha\lambda_i}$$

# Convergence SGD



$$e^{-k/\tau_i} \cdot (\underline{D}[0])_i \approx (1 - 2\alpha\lambda_i)^k \cdot (\underline{D}[0])_i \Rightarrow$$

Average behavior:  $\tau_{av,i} = \frac{-1}{\ln(1 - 2\alpha\lambda_i)}$   $\Rightarrow$  For small  $\alpha$

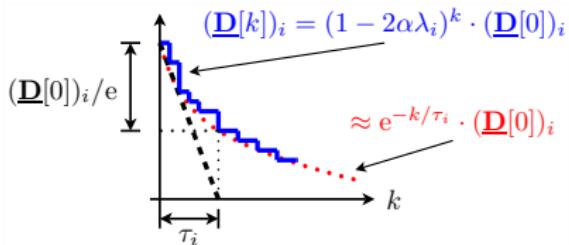
$$\tau_{av,i} \approx \frac{1}{2\alpha\lambda_i}$$

*Note:*

Overall time constant depends on eigenvalue spread

$\Gamma_x = \lambda_{max}/\lambda_{min}$ . Thus, the larger  $\Gamma_x$  the longer it takes for adaptation.

# Convergence SGD



$$e^{-k/\tau_i} \cdot (\underline{D}[0])_i \approx (1 - 2\alpha\lambda_i)^k \cdot (\underline{D}[0])_i \Rightarrow$$

Average behavior:  $\tau_{av,i} = \frac{-1}{\ln(1 - 2\alpha\lambda_i)}$   $\Rightarrow$  For small  $\alpha$

$$\tau_{av,i} \approx \frac{1}{2\alpha\lambda_i}$$

*Note:*

Overall time constant depends on eigenvalue spread

$\Gamma_x = \lambda_{max}/\lambda_{min}$ . Thus, the larger  $\Gamma_x$  the longer it takes for adaptation.

**Q:** What happens for white noise?

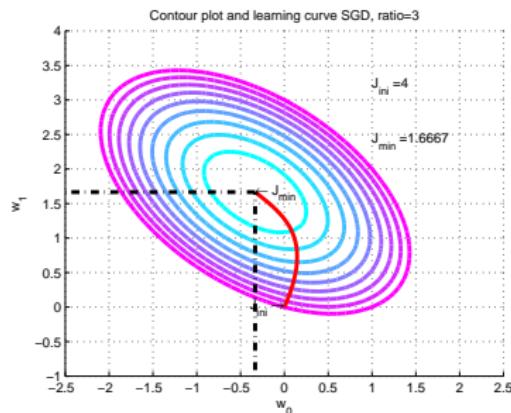
# Convergence SGD

Example with  $\Gamma_x = \lambda_{max}/\lambda_{min} = 3$

# Convergence SGD

Example with  $\Gamma_x = \lambda_{max}/\lambda_{min} = 3$

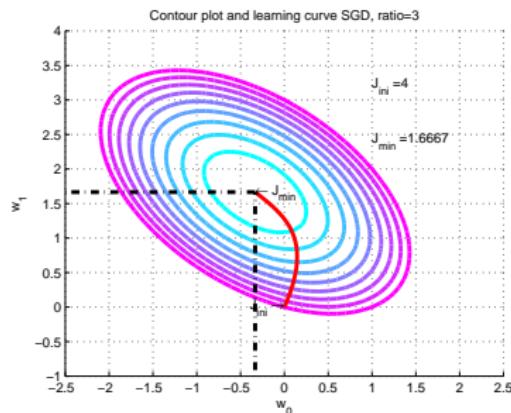
## Learning curve in contour plot $J$



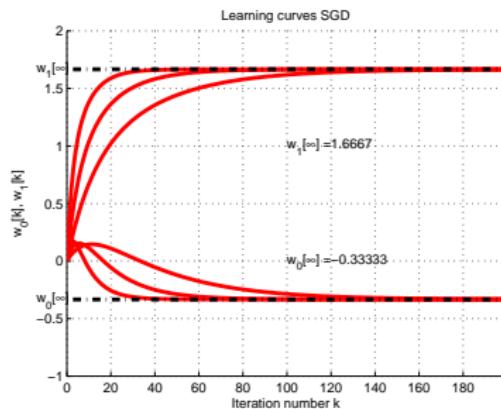
# Convergence SGD

Example with  $\Gamma_x = \lambda_{max}/\lambda_{min} = 3$

## Learning curve in contour plot $J$



## Learning curves for different $\alpha$



## Least Mean Square (LMS)

**Motivation:** SGD not practical. Gradient assumes **known**  $R_x$  and  $r_{ex}$

# Least Mean Square (LMS)

**Motivation:** SGD not practical. Gradient assumes **known**  $R_x$  and  $r_{ex}$

**LMS principle:** Use instantaneous estimate of gradient:

$$\begin{aligned}\hat{\nabla}[k] &= -2(e[k]\underline{x}[k] - \underline{x}[k]\underline{x}^t[k]\underline{w}[k]) \\ &= -2\underline{x}[k](e[k] - \underline{x}^t[k]\underline{w}[k]) = -2\underline{x}[k]r[k]\end{aligned}$$

# Least Mean Square (LMS)

**Motivation:** SGD not practical. Gradient assumes **known**  $R_x$  and  $r_{ex}$

**LMS principle:** Use instantaneous estimate of gradient:

$$\begin{aligned}\hat{\nabla}[k] &= -2(e[k]\underline{x}[k] - \underline{x}[k]\underline{x}^t[k]\underline{w}[k]) \\ &= -2\underline{x}[k](e[k] - \underline{x}^t[k]\underline{w}[k]) = -2\underline{x}[k]r[k]\end{aligned}$$

With  $\underline{w} \doteq \underline{w} - \hat{\nabla} \Rightarrow$  LMS algorithm (Widrow, 1975):

$$k = 0 : \underline{w}[0] = \underline{0} \text{ (usually)}$$

$$k > 0 : \hat{e}[k] = \underline{w}^t[k] \cdot \underline{x}[k]$$

$$r[k] = e[k] - \hat{e}[k]$$

$$\underline{w}[k+1] = \underline{w}[k] + 2\alpha \underline{x}[k]r[k]$$

# Least Mean Square (LMS)

**Motivation:** SGD not practical. Gradient assumes **known**  $R_x$  and  $r_{ex}$

**LMS principle:** Use instantaneous estimate of gradient:

$$\begin{aligned}\hat{\nabla}[k] &= -2(e[k]\underline{x}[k] - \underline{x}[k]\underline{x}^t[k]\underline{w}[k]) \\ &= -2\underline{x}[k](e[k] - \underline{x}^t[k]\underline{w}[k]) = -2\underline{x}[k]r[k]\end{aligned}$$

With  $\underline{w} \doteq \underline{w} - \hat{\nabla} \Rightarrow$  LMS algorithm (Widrow, 1975):

$$k = 0 : \underline{w}[0] = \underline{0} \text{ (usually)}$$

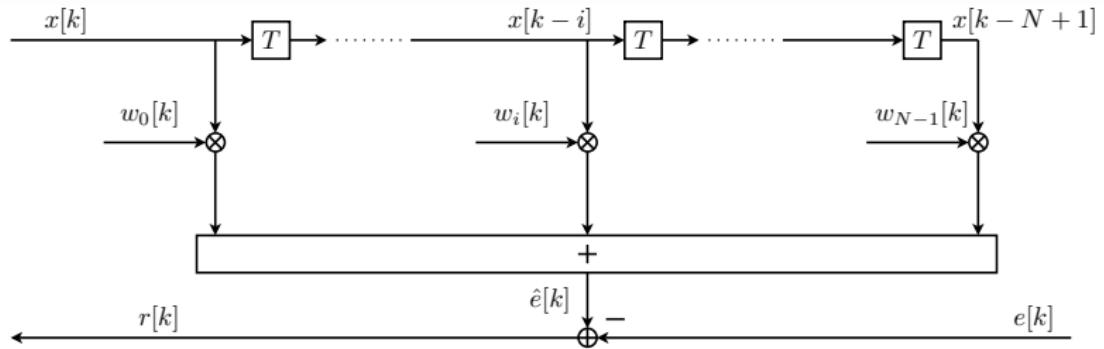
$$k > 0 : \hat{e}[k] = \underline{w}^t[k] \cdot \underline{x}[k]$$

$$r[k] = e[k] - \hat{e}[k]$$

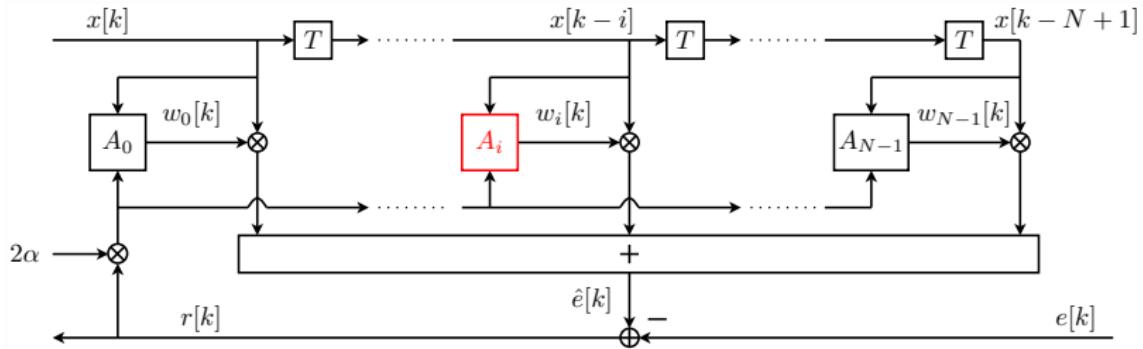
$$\underline{w}[k+1] = \underline{w}[k] + 2\alpha \underline{x}[k]r[k]$$

Note:  $\underline{w}^t[k] \cdot \underline{x}[k]$  is "convolution" and  $\underline{x}[k]r[k]$  "correlation"

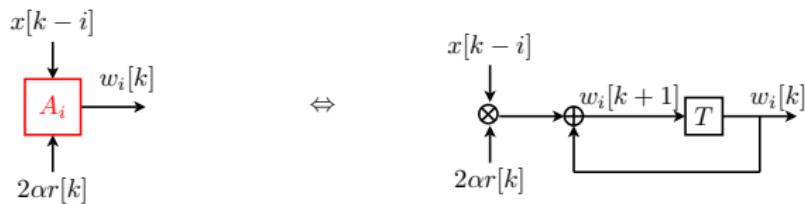
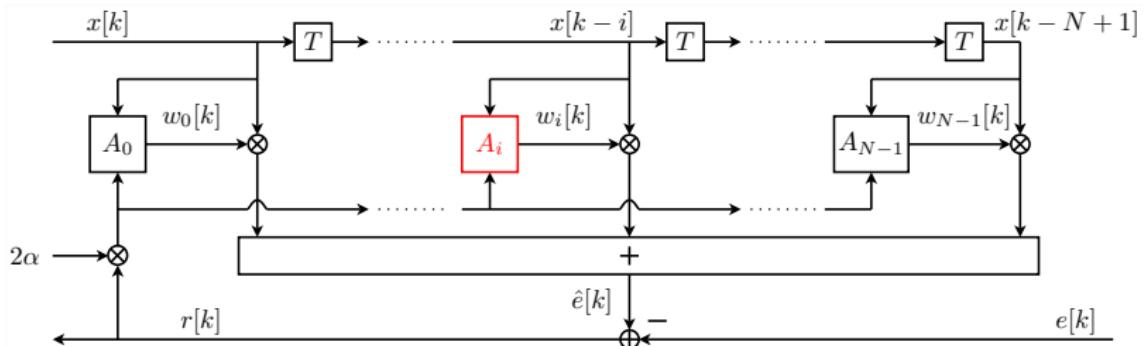
## Realization scheme LMS algorithm:



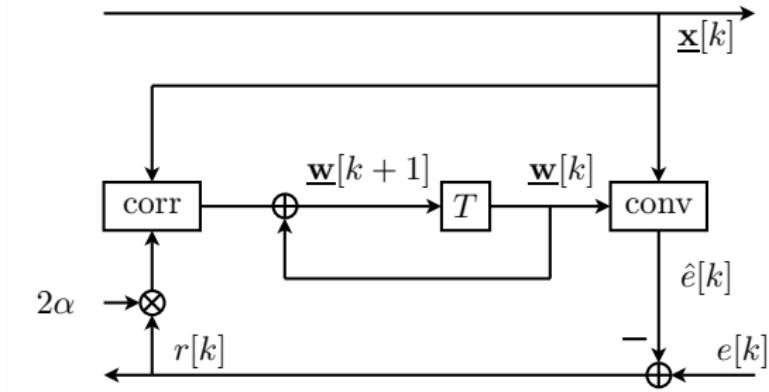
## Realization scheme LMS algorithm:



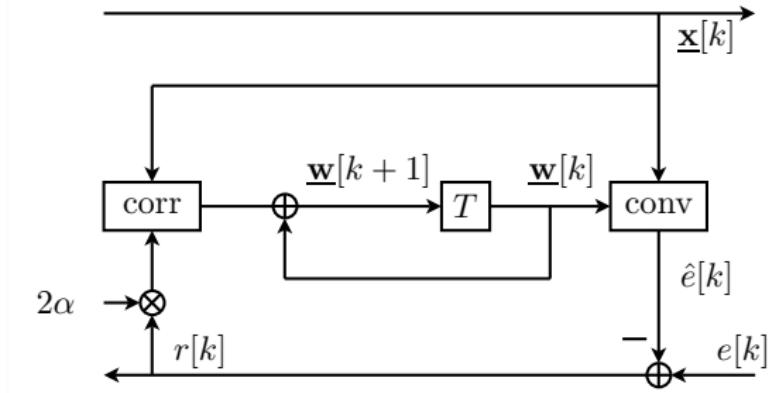
## Realization scheme LMS algorithm:



## Simplified realization scheme LMS algorithm:



## Simplified realization scheme LMS algorithm:



Notes:

- ▶ Simple, robust algorithm, complexity  $O(2N)$
- ▶ LMS tries to "decorrelate" signals  $x$  and  $r$
- ▶ In contrast to SGD: Weights fluctuate around optimal values

## Two LMS variants

## Two LMS variants

- ▶ **NLMS:** LMS with normalization by  $\sigma_x^2 = E\{x^2[k]\}$ :

$$\underline{\mathbf{w}}[k+1] = \underline{\mathbf{w}}[k] + \frac{2\alpha}{\sigma_x^2} \underline{\mathbf{x}}[k] r[k]$$

## Two LMS variants

- ▶ **NLMS:** LMS with normalization by  $\sigma_x^2 = E\{x^2[k]\}$ :

$$\underline{\mathbf{w}}[k+1] = \underline{\mathbf{w}}[k] + \frac{2\alpha}{\sigma_x^2} \underline{\mathbf{x}}[k] r[k]$$

In practice  $\hat{\sigma}_x^2[k] \Rightarrow$  time-varying step size. E.g.:

- ▶  $\hat{\sigma}_x^2[k] = \beta \hat{\sigma}_x^2[k-1] + (1 - \beta) \frac{\underline{\mathbf{x}}^t[k] \underline{\mathbf{x}}[k]}{N}$  with  $0 < \beta < 1$
- ▶  $\hat{\sigma}_x^2[k] = \frac{\underline{\mathbf{x}}^t[k] \underline{\mathbf{x}}[k]}{N} + \epsilon$  with  $\epsilon$  some small constant

## Two LMS variants

- ▶ **NLMS:** LMS with normalization by  $\sigma_x^2 = E\{x^2[k]\}$ :

$$\underline{w}[k+1] = \underline{w}[k] + \frac{2\alpha}{\sigma_x^2} \underline{x}[k] r[k]$$

In practice  $\hat{\sigma}_x^2[k] \Rightarrow$  time-varying step size. E.g.:

- ▶  $\hat{\sigma}_x^2[k] = \beta \hat{\sigma}_x^2[k-1] + (1 - \beta) \frac{\underline{x}^t[k] \underline{x}[k]}{N}$  with  $0 < \beta < 1$
- ▶  $\hat{\sigma}_x^2[k] = \frac{\underline{x}^t[k] \underline{x}[k]}{N} + \epsilon$  with  $\epsilon$  some small constant

- ▶ **Complex LMS:** LMS for complex signals and weights:

$$\underline{w}[k+1] = \underline{w}[k] + 2\alpha \underline{x}[k] r^*[k]$$

# Newton

Convergence gradient based algorithms relies on coloration input:

$$\underline{\nabla} = -2(\underline{r}_{ex} - \mathbf{R}_x \underline{w}[k])$$

# Newton

Convergence gradient based algorithms relies on coloration input:

$$\underline{\nabla} = -2(\underline{r}_{ex} - \mathbf{R}_x \underline{w}[k])$$

**Solution Newton:**  $\underline{w}[k + 1] = \underline{w}[k] - \alpha \mathbf{R}_x^{-1} \underline{\nabla} \Rightarrow$

$$\underline{w}[k + 1] = \underline{w}[k] + 2\alpha \mathbf{R}_x^{-1} \cdot (\underline{r}_{ex} - \mathbf{R}_x \underline{w}[k])$$

# Newton

Convergence gradient based algorithms relies on coloration input:

$$\underline{\nabla} = -2(\underline{r}_{ex} - \mathbf{R}_x \underline{w}[k])$$

**Solution Newton:**  $\underline{w}[k + 1] = \underline{w}[k] - \alpha \mathbf{R}_x^{-1} \underline{\nabla} \Rightarrow$

$$\underline{w}[k + 1] = \underline{w}[k] + 2\alpha \mathbf{R}_x^{-1} \cdot (\underline{r}_{ex} - \mathbf{R}_x \underline{w}[k])$$

Convergence Newton:

$$\underline{d}[k+1] = (\mathbf{I} - 2\alpha \mathbf{R}_x^{-1} \mathbf{R}_x) \underline{d}[k] = (1 - 2\alpha) \underline{d}[k] \Rightarrow \text{Convergence } 0 < \alpha < \frac{1}{2}$$

# Newton

Convergence gradient based algorithms relies on coloration input:

$$\underline{\nabla} = -2(\underline{r}_{ex} - \mathbf{R}_x \underline{w}[k])$$

**Solution Newton:**  $\underline{w}[k+1] = \underline{w}[k] - \alpha \mathbf{R}_x^{-1} \underline{\nabla} \Rightarrow$

$$\underline{w}[k+1] = \underline{w}[k] + 2\alpha \mathbf{R}_x^{-1} \cdot (\underline{r}_{ex} - \mathbf{R}_x \underline{w}[k])$$

Convergence Newton:

$$\underline{d}[k+1] = (\mathbf{I} - 2\alpha \mathbf{R}_x^{-1} \mathbf{R}_x) \underline{d}[k] = (1-2\alpha) \underline{d}[k] \Rightarrow \text{Convergence } 0 < \alpha < 1$$

Notes:

- ▶  $\mathbf{R}_x^{-1}$  causes whitening of input process

# Newton

Convergence gradient based algorithms relies on coloration input:

$$\underline{\nabla} = -2(\underline{r}_{ex} - \mathbf{R}_x \underline{w}[k])$$

**Solution Newton:**  $\underline{w}[k+1] = \underline{w}[k] - \alpha \mathbf{R}_x^{-1} \underline{\nabla} \Rightarrow$

$$\underline{w}[k+1] = \underline{w}[k] + 2\alpha \mathbf{R}_x^{-1} \cdot (\underline{r}_{ex} - \mathbf{R}_x \underline{w}[k])$$

Convergence Newton:

$$\underline{d}[k+1] = (\mathbf{I} - 2\alpha \mathbf{R}_x^{-1} \mathbf{R}_x) \underline{d}[k] = (1-2\alpha) \underline{d}[k] \Rightarrow \text{Convergence } 0 < \alpha < \frac{1}{2}$$

Notes:

- ▶  $\mathbf{R}_x^{-1}$  causes whitening of input process
- ▶ All weights have same convergence (in contrast to LMS, SGD)

# Newton

Convergence gradient based algorithms relies on coloration input:

$$\underline{\nabla} = -2(\underline{r}_{ex} - \mathbf{R}_x \underline{w}[k])$$

**Solution Newton:**  $\underline{w}[k+1] = \underline{w}[k] - \alpha \mathbf{R}_x^{-1} \underline{\nabla} \Rightarrow$

$$\underline{w}[k+1] = \underline{w}[k] + 2\alpha \mathbf{R}_x^{-1} \cdot (\underline{r}_{ex} - \mathbf{R}_x \underline{w}[k])$$

Convergence Newton:

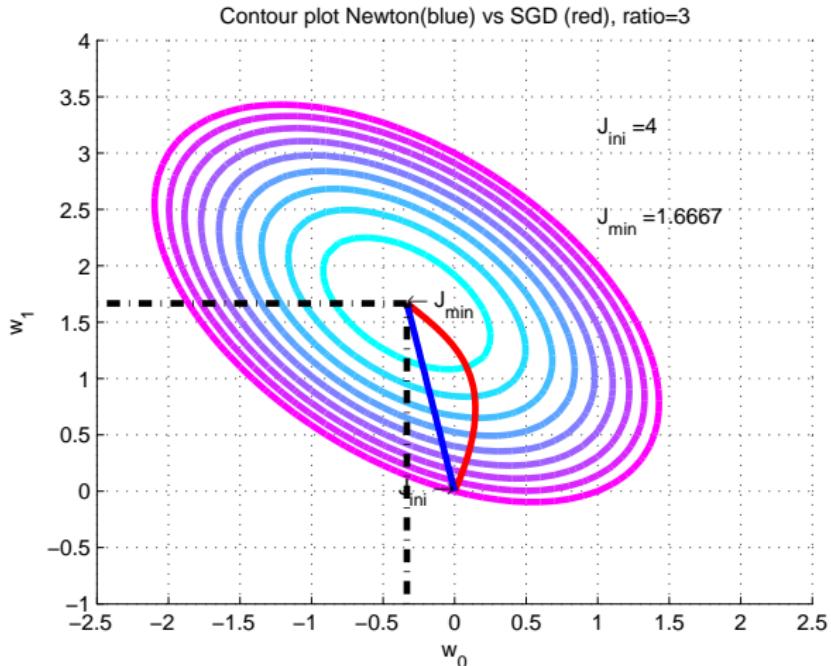
$$\underline{d}[k+1] = (\mathbf{I} - 2\alpha \mathbf{R}_x^{-1} \mathbf{R}_x) \underline{d}[k] = (1-2\alpha) \underline{d}[k] \Rightarrow \text{Convergence } 0 < \alpha < \frac{1}{2}$$

Notes:

- ▶  $\mathbf{R}_x^{-1}$  causes whitening of input process
- ▶ All weights have same convergence (in contrast to LMS, SGD)
- ▶ Newton  $\equiv$  SGD with white noise input!

# Newton

## Learning curves in contour plot: Newton vs. SGD



## Newton: another view

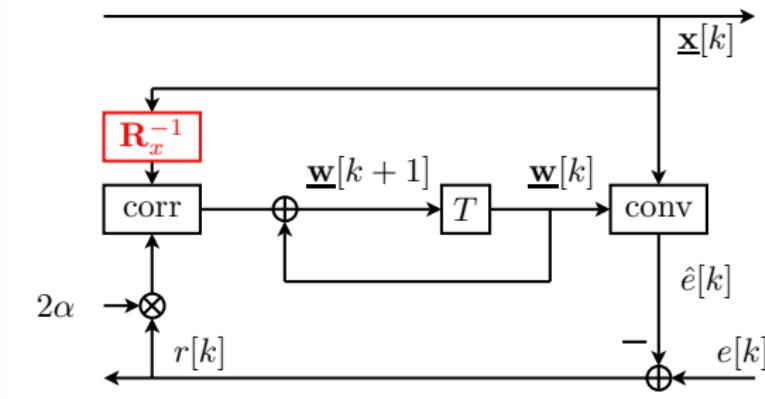
Replace  $\underline{\triangledown}$  by  $\hat{\underline{\triangledown}}_{LMS} = \underline{x}[k]r[k] \Rightarrow \text{LMS/Newton:}$

$$\underline{w}[k+1] = \underline{w}[k] + 2\alpha \mathbf{R}_x^{-1} \underline{x}[k]r[k]$$

## Newton: another view

Replace  $\underline{\triangledown}$  by  $\hat{\underline{\triangledown}}_{LMS} = \underline{x}[k]r[k] \Rightarrow \text{LMS/Newton:}$

$$\underline{w}[k + 1] = \underline{w}[k] + 2\alpha \underline{R}_x^{-1} \underline{x}[k]r[k]$$



# Newton: Practical problem

Autocorrelation matrix  $R_x$ :

# Newton: Practical problem

Autocorrelation matrix  $R_x$ :

- ▶ (In general) not known in advance
- ▶ May change during time (non-stationary process)
- ▶ Inversion is expensive (many MIPS)

# Newton: Practical problem

Autocorrelation matrix  $R_x$ :

- ▶ (In general) not known in advance
- ▶ May change during time (non-stationary process)
- ▶ Inversion is expensive (many MIPS)
  - ⇒ Complexity Newton algorithm huge
  - ⇒ Need for efficient solution with estimate of  $R_x$
  - ⇒ Different algorithms, e.g. RLS, FDAF, etc.

## Recursive Least Squares (RLS)

For data block length  $L$  fixed, Least Squares problem becomes:

$$\min_{\underline{w}[k]} |\underline{e}[k] - \mathbf{X}[k] \cdot \underline{w}[k]|^2 \Rightarrow \underline{w}_{LS}[k] = (\mathbf{X}^t[k] \mathbf{X}[k])^{-1} (\mathbf{X}^t[k] \underline{e}[k])$$

## Recursive Least Squares (RLS)

For data block length  $L$  fixed, Least Squares problem becomes:

$$\min_{\underline{w}[k]} |\underline{e}[k] - \mathbf{X}[k] \cdot \underline{w}[k]|^2 \Rightarrow \underline{w}_{LS}[k] = (\mathbf{X}^t[k] \mathbf{X}[k])^{-1} (\mathbf{X}^t[k] \underline{e}[k])$$

**RLS:** Find efficient recursive solution for LS problem from  
 $k \rightarrow k + 1$

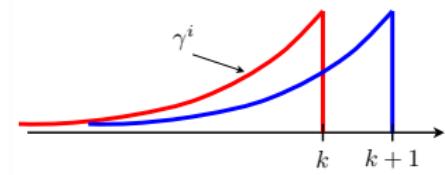
# Recursive Least Squares (RLS)

For data block length  $L$  fixed, Least Squares problem becomes:

$$\min_{\underline{w}[k]} |\underline{e}[k] - \mathbf{X}[k] \cdot \underline{w}[k]|^2 \Rightarrow \underline{w}_{LS}[k] = (\mathbf{X}^t[k] \mathbf{X}[k])^{-1} (\mathbf{X}^t[k] \underline{e}[k])$$

**RLS:** Find efficient recursive solution for LS problem from  $k \rightarrow k + 1$

Use exponential sliding window: Scale down data by factor  $\gamma$



Forgetting factor :  $0 < \gamma < 1$

'Memory' :  $\frac{1}{1 - \gamma}$

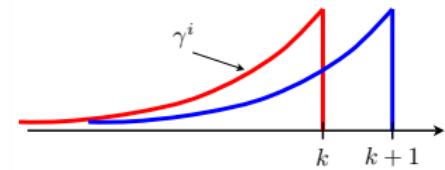
# Recursive Least Squares (RLS)

For data block length  $L$  fixed, Least Squares problem becomes:

$$\min_{\underline{w}[k]} |\underline{e}[k] - \underline{X}[k] \cdot \underline{w}[k]|^2 \Rightarrow \underline{w}_{LS}[k] = (\underline{X}^t[k] \underline{X}[k])^{-1} (\underline{X}^t[k] \underline{e}[k])$$

**RLS:** Find efficient recursive solution for LS problem from  $k \rightarrow k + 1$

Use exponential sliding window: Scale down data by factor  $\gamma$



Forgetting factor :  $0 < \gamma < 1$

'Memory' :  $\frac{1}{1-\gamma}$

$$\underline{X}[k] = \begin{pmatrix} \gamma^0 \underline{x}^t[k] \\ \dots \\ \gamma^i \underline{x}^t[k-i] \\ \dots \\ \gamma^k \underline{x}^t[0] \end{pmatrix} \text{ and } \underline{e}[k] = \begin{pmatrix} \gamma^0 e[k] \\ \dots \\ \gamma^i e[k-i] \\ \dots \\ \gamma^k e[0] \end{pmatrix}$$

# RLS algorithm

## RLS algorithm

**Initialization:**  $\underline{r}_{ex}[0] = \underline{0}$  ;  $\overline{R}_x^{-1}[0] = \delta^{-1}I$  with  $\delta$  small

# RLS algorithm

Initialization:  $\underline{r}_{ex}[0] = \underline{0}$  ;  $\overline{R}_x^{-1}[0] = \delta^{-1}I$  with  $\delta$  small

For  $k \geq 0$ :

## RLS algorithm

Initialization:  $\underline{r}_{ex}[0] = \underline{0}$  ;  $\overline{R}_x^{-1}[0] = \delta^{-1}I$  with  $\delta$  small

For  $k \geq 0$ :

$$\overline{R}_x^{-1}[k+1] = \gamma^{-2} \left( \overline{R}_x^{-1}[k] - \underline{g}[k+1] \cdot \underline{x}^t[k+1] \overline{R}_x^{-1}[k] \right)$$

## RLS algorithm

Initialization:  $\underline{r}_{ex}[0] = \underline{0}$  ;  $\bar{R}_x^{-1}[0] = \delta^{-1}I$  with  $\delta$  small

$$\text{For } k \geq \underline{g}[k+1] \quad = \quad \frac{\bar{R}_x^{-1}[k]\underline{x}[k+1]}{\gamma^2 + \underline{x}^t[k+1]\bar{R}_x^{-1}[k]\underline{x}[k+1]}$$

$$\bar{R}_x^{-1}[k+1] \quad = \quad \gamma^{-2} \left( \bar{R}_x^{-1}[k] - \underline{g}[k+1] \cdot \underline{x}^t[k+1]\bar{R}_x^{-1}[k] \right)$$

## RLS algorithm

Initialization:  $\underline{r}_{ex}[0] = \underline{0}$  ;  $\bar{R}_x^{-1}[0] = \delta^{-1}I$  with  $\delta$  small

$$\text{For } k \geq \underline{g}[k+1] \quad = \quad \frac{\bar{R}_x^{-1}[k]\underline{x}[k+1]}{\gamma^2 + \underline{x}^t[k+1]\bar{R}_x^{-1}[k]\underline{x}[k+1]}$$

$$\bar{R}_x^{-1}[k+1] \quad = \quad \gamma^{-2} \left( \bar{R}_x^{-1}[k] - \underline{g}[k+1] \cdot \underline{x}^t[k+1] \bar{R}_x^{-1}[k] \right)$$

$$\underline{r}_{ex}[k+1] \quad = \quad \gamma^2 \underline{r}_{ex}[k] + \underline{x}[k+1] \cdot e[k+1]$$

## RLS algorithm

Initialization:  $\underline{r}_{ex}[0] = \underline{0}$  ;  $\overline{R}_x^{-1}[0] = \delta^{-1}I$  with  $\delta$  small

$$\text{For } k \geq 0, \underline{g}[k+1] = \frac{\overline{R}_x^{-1}[k]\underline{x}[k+1]}{\gamma^2 + \underline{x}^t[k+1]\overline{R}_x^{-1}[k]\underline{x}[k+1]}$$

$$\overline{R}_x^{-1}[k+1] = \gamma^{-2} \left( \overline{R}_x^{-1}[k] - \underline{g}[k+1] \cdot \underline{x}^t[k+1] \overline{R}_x^{-1}[k] \right)$$

$$\underline{r}_{ex}[k+1] = \gamma^2 \underline{r}_{ex}[k] + \underline{x}[k+1] \cdot e[k+1]$$

$$\underline{w}[k+1] = \overline{R}_x^{-1}[k+1] \cdot \underline{r}_{ex}[k+1]$$

# RLS algorithm

Initialization:  $\underline{r}_{ex}[0] = \underline{0}$  ;  $\overline{R}_x^{-1}[0] = \delta^{-1}I$  with  $\delta$  small

For  $k \geq 0$ :

$$\underline{g}[k+1] = \frac{\overline{R}_x^{-1}[k]\underline{x}[k+1]}{\gamma^2 + \underline{x}^t[k+1]\overline{R}_x^{-1}[k]\underline{x}[k+1]}$$

$$\overline{R}_x^{-1}[k+1] = \gamma^{-2} \left( \overline{R}_x^{-1}[k] - \underline{g}[k+1] \cdot \underline{x}^t[k+1] \overline{R}_x^{-1}[k] \right)$$

$$\underline{r}_{ex}[k+1] = \gamma^2 \underline{r}_{ex}[k] + \underline{x}[k+1] \cdot \underline{e}[k+1]$$

$$\underline{w}[k+1] = \overline{R}_x^{-1}[k+1] \cdot \underline{r}_{ex}[k+1]$$

Notes:

- ▶  $\underline{w}[\infty] \rightarrow \underline{w}_o$

## RLS algorithm

Initialization:  $\underline{r}_{ex}[0] = \underline{0}$  ;  $\bar{R}_x^{-1}[0] = \delta^{-1}I$  with  $\delta$  small

$$\text{For } k \geq \underline{g}[k+1] \quad = \quad \frac{\bar{R}_x^{-1}[k]\underline{x}[k+1]}{\gamma^2 + \underline{x}^t[k+1]\bar{R}_x^{-1}[k]\underline{x}[k+1]}$$

$$\bar{R}_x^{-1}[k+1] \quad = \quad \gamma^{-2} \left( \bar{R}_x^{-1}[k] - \underline{g}[k+1] \cdot \underline{x}^t[k+1] \bar{R}_x^{-1}[k] \right)$$

$$\underline{r}_{ex}[k+1] \quad = \quad \gamma^2 \underline{r}_{ex}[k] + \underline{x}[k+1] \cdot e[k+1]$$

$$\underline{w}[k+1] \quad = \quad \bar{R}_x^{-1}[k+1] \cdot \underline{r}_{ex}[k+1]$$

Notes:

- ▶  $\underline{w}[\infty] \rightarrow \underline{w}_o$
- ▶ Complexity  $O(N^2)$  per time update

# RLS algorithm

Initialization:  $\underline{r}_{ex}[0] = \underline{0}$  ;  $\bar{R}_x^{-1}[0] = \delta^{-1}I$  with  $\delta$  small

$$\text{For } k \geq 0, \underline{g}[k+1] = \frac{\bar{R}_x^{-1}[k]\underline{x}[k+1]}{\gamma^2 + \underline{x}^t[k+1]\bar{R}_x^{-1}[k]\underline{x}[k+1]}$$

$$\bar{R}_x^{-1}[k+1] = \gamma^{-2} \left( \bar{R}_x^{-1}[k] - \underline{g}[k+1] \cdot \underline{x}^t[k+1] \bar{R}_x^{-1}[k] \right)$$

$$\underline{r}_{ex}[k+1] = \gamma^2 \underline{r}_{ex}[k] + \underline{x}[k+1] \cdot e[k+1]$$

$$\underline{w}[k+1] = \bar{R}_x^{-1}[k+1] \cdot \underline{r}_{ex}[k+1]$$

Notes:

- ▶  $\underline{w}[\infty] \rightarrow \underline{w}_o$
- ▶ Complexity  $O(N^2)$  per time update
- ▶ Window length increases when time increases!

## RLS algorithm

Initialization:  $\underline{r}_{ex}[0] = \underline{0}$  ;  $\bar{R}_x^{-1}[0] = \delta^{-1}I$  with  $\delta$  small

$$\text{For } k \geq 0, \underline{g}[k+1] = \frac{\bar{R}_x^{-1}[k]\underline{x}[k+1]}{\gamma^2 + \underline{x}^t[k+1]\bar{R}_x^{-1}[k]\underline{x}[k+1]}$$

$$\bar{R}_x^{-1}[k+1] = \gamma^{-2} \left( \bar{R}_x^{-1}[k] - \underline{g}[k+1] \cdot \underline{x}^t[k+1] \bar{R}_x^{-1}[k] \right)$$

$$\underline{r}_{ex}[k+1] = \gamma^2 \underline{r}_{ex}[k] + \underline{x}[k+1] \cdot e[k+1]$$

$$\underline{w}[k+1] = \bar{R}_x^{-1}[k+1] \cdot \underline{r}_{ex}[k+1]$$

Notes:

- ▶  $\underline{w}[\infty] \rightarrow \underline{w}_o$
- ▶ Complexity  $O(N^2)$  per time update
- ▶ Window length increases when time increases!
- ▶ Exhibits unstable roundoff error accumulation

## RLS algorithm

Initialization:  $\underline{r}_{ex}[0] = \underline{0}$  ;  $\overline{R}_x^{-1}[0] = \delta^{-1}I$  with  $\delta$  small

$$\text{For } k \geq 0, \underline{g}[k+1] = \frac{\overline{R}_x^{-1}[k]\underline{x}[k+1]}{\gamma^2 + \underline{x}^t[k+1]\overline{R}_x^{-1}[k]\underline{x}[k+1]}$$

$$\overline{R}_x^{-1}[k+1] = \gamma^{-2} \left( \overline{R}_x^{-1}[k] - \underline{g}[k+1] \cdot \underline{x}^t[k+1] \overline{R}_x^{-1}[k] \right)$$

$$\underline{r}_{ex}[k+1] = \gamma^2 \underline{r}_{ex}[k] + \underline{x}[k+1] \cdot e[k+1]$$

$$\underline{w}[k+1] = \overline{R}_x^{-1}[k+1] \cdot \underline{r}_{ex}[k+1]$$

Notes:

- ▶  $\underline{w}[\infty] \rightarrow \underline{w}_o$
- ▶ Complexity  $O(N^2)$  per time update
- ▶ Window length increases when time increases!
- ▶ Exhibits unstable roundoff error accumulation
- ▶ RLS is basis for many practical algorithms

## RLS algorithm

Initialization:  $\underline{r}_{ex}[0] = \underline{0}$  ;  $\overline{R}_x^{-1}[0] = \delta^{-1}I$  with  $\delta$  small

$$\text{For } k \geq 0, \underline{g}[k+1] = \frac{\overline{R}_x^{-1}[k]\underline{x}[k+1]}{\gamma^2 + \underline{x}^t[k+1]\overline{R}_x^{-1}[k]\underline{x}[k+1]}$$

$$\overline{R}_x^{-1}[k+1] = \gamma^{-2} \left( \overline{R}_x^{-1}[k] - \underline{g}[k+1] \cdot \underline{x}^t[k+1] \overline{R}_x^{-1}[k] \right)$$

$$\underline{r}_{ex}[k+1] = \gamma^2 \underline{r}_{ex}[k] + \underline{x}[k+1] \cdot e[k+1]$$

$$\underline{w}[k+1] = \overline{R}_x^{-1}[k+1] \cdot \underline{r}_{ex}[k+1]$$

Notes:

- ▶  $\underline{w}[\infty] \rightarrow \underline{w}_o$
- ▶ Complexity  $O(N^2)$  per time update
- ▶ Window length increases when time increases!
- ▶ Exhibits unstable roundoff error accumulation
- ▶ RLS is basis for many practical algorithms
- ▶ Decorrelation takes place in algorithm

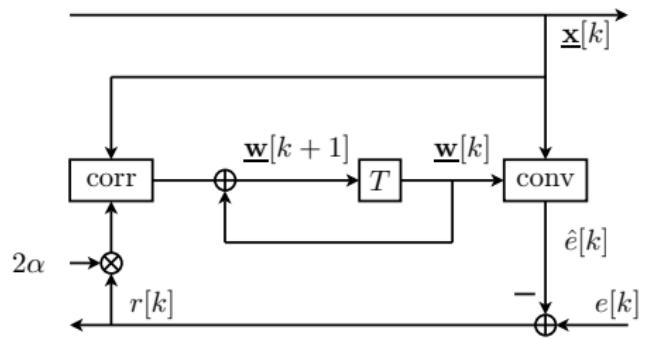
# Frequency Domain Adaptive Filter (FDAF)

**FDAF:** Alternative for LMS/Newton and RLS

# Frequency Domain Adaptive Filter (FDAF)

**FDAF:** Alternative for LMS/Newton and RLS

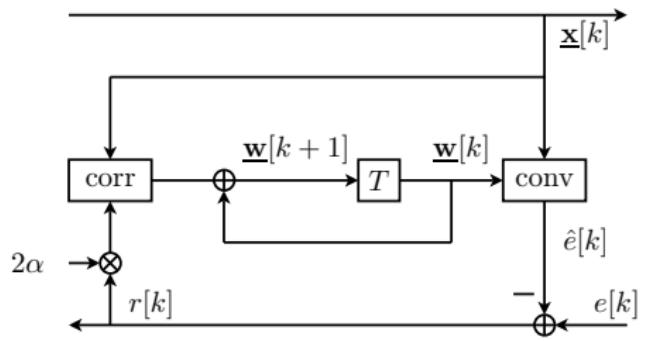
*First step of derivation:* Translate LMS to frequency domain



# Frequency Domain Adaptive Filter (FDAF)

**FDAF:** Alternative for LMS/Newton and RLS

*First step of derivation:* Translate LMS to frequency domain



LMS weight update:

$$\underline{w}[k + 1] = \underline{w}[k] + 2\alpha \underline{x}[k] r[k]$$

Filter output:

$$\hat{e}[k] = \underline{x}^t[k] \cdot \underline{w}[k]$$

# FDAF

*Transform vectors to frequency domain:*

# FDAF

*Transform vectors to frequency domain:*

$$\mathbf{F} \cdot \underline{x}[k] = \underline{X}[k] = (X_0[k], X_1[k], \dots, X_{N-1}[k])^t$$

$$\mathbf{F}^{-1} \cdot \underline{w}[k] = \underline{W}[k] = (W_0[k], W_1[k], \dots, W_{N-1}[k])^t$$

with DFT matrix  $\mathbf{F}$ , which has properties:  $\mathbf{F} = \mathbf{F}^t$  and  $\mathbf{F}^{-1} = \frac{1}{N}\mathbf{F}^*$

# FDAF

Transform vectors to frequency domain:

$$\mathbf{F} \cdot \underline{x}[k] = \underline{X}[k] = (X_0[k], X_1[k], \dots, X_{N-1}[k])^t$$

$$\mathbf{F}^{-1} \cdot \underline{w}[k] = \underline{W}[k] = (W_0[k], W_1[k], \dots, W_{N-1}[k])^t$$

with DFT matrix  $\mathbf{F}$ , which has properties:  $\mathbf{F} = \mathbf{F}^t$  and  $\mathbf{F}^{-1} = \frac{1}{N}\mathbf{F}^*$

Apply filter operation in frequency domain:

$$\hat{e}[k] = \sum_{i=0}^{N-1} x[k-i] \cdot w_i[k] = \underline{x}^t \cdot \underline{w}[k]$$

# FDAF

Transform vectors to frequency domain:

$$\mathbf{F} \cdot \underline{x}[k] = \underline{X}[k] = (X_0[k], X_1[k], \dots, X_{N-1}[k])^t$$

$$\mathbf{F}^{-1} \cdot \underline{w}[k] = \underline{W}[k] = (W_0[k], W_1[k], \dots, W_{N-1}[k])^t$$

with DFT matrix  $\mathbf{F}$ , which has properties:  $\mathbf{F} = \mathbf{F}^t$  and  $\mathbf{F}^{-1} = \frac{1}{N}\mathbf{F}^*$

Apply filter operation in frequency domain:

$$\begin{aligned}\hat{e}[k] &= \sum_{i=0}^{N-1} x[k-i] \cdot w_i[k] = \underline{x}^t \cdot \underline{w}[k] \\ &= \underline{x}^t[k](\mathbf{F} \cdot \mathbf{F}^{-1})\underline{w}[k] = (\mathbf{F}\underline{x}[k])^t \cdot (\mathbf{F}^{-1}\underline{w}[k])\end{aligned}$$

# FDAF

Transform vectors to frequency domain:

$$\mathbf{F} \cdot \underline{x}[k] = \underline{X}[k] = (X_0[k], X_1[k], \dots, X_{N-1}[k])^t$$

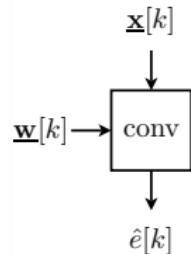
$$\mathbf{F}^{-1} \cdot \underline{w}[k] = \underline{W}[k] = (W_0[k], W_1[k], \dots, W_{N-1}[k])^t$$

with DFT matrix  $\mathbf{F}$ , which has properties:  $\mathbf{F} = \mathbf{F}^t$  and  $\mathbf{F}^{-1} = \frac{1}{N}\mathbf{F}^*$

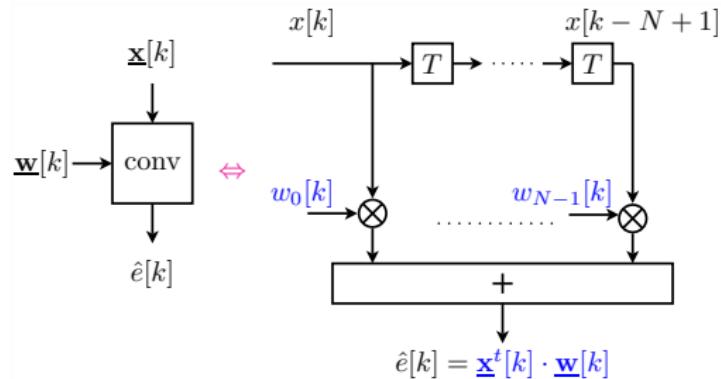
Apply filter operation in frequency domain:

$$\begin{aligned}\hat{e}[k] &= \sum_{i=0}^{N-1} x[k-i] \cdot w_i[k] = \underline{x}^t \cdot \underline{w}[k] \\ &= \underline{x}^t[k](\mathbf{F} \cdot \mathbf{F}^{-1})\underline{w}[k] = (\mathbf{F}\underline{x}[k])^t \cdot (\mathbf{F}^{-1}\underline{w}[k]) \\ &= \underline{X}^t[k] \cdot \underline{W}[k] = \sum_{l=0}^{N-1} X_l[k] W_l[k]\end{aligned}$$

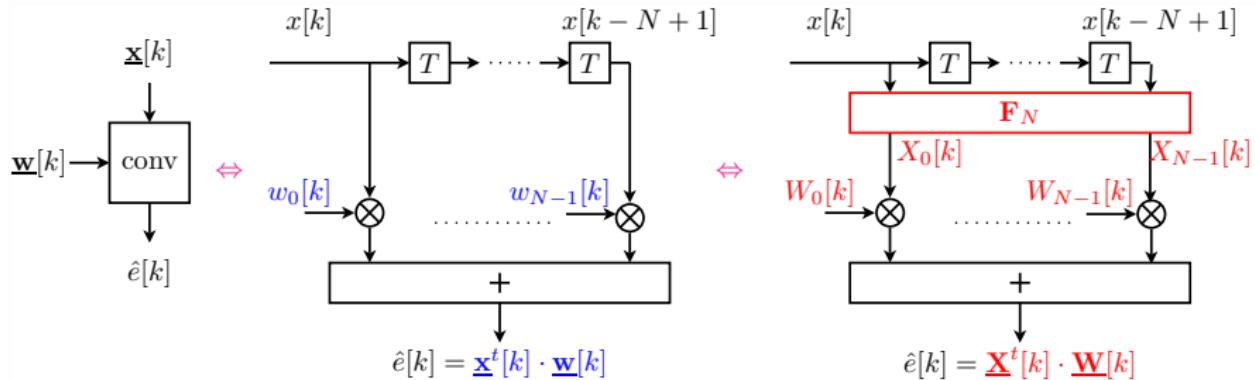
# FDAF



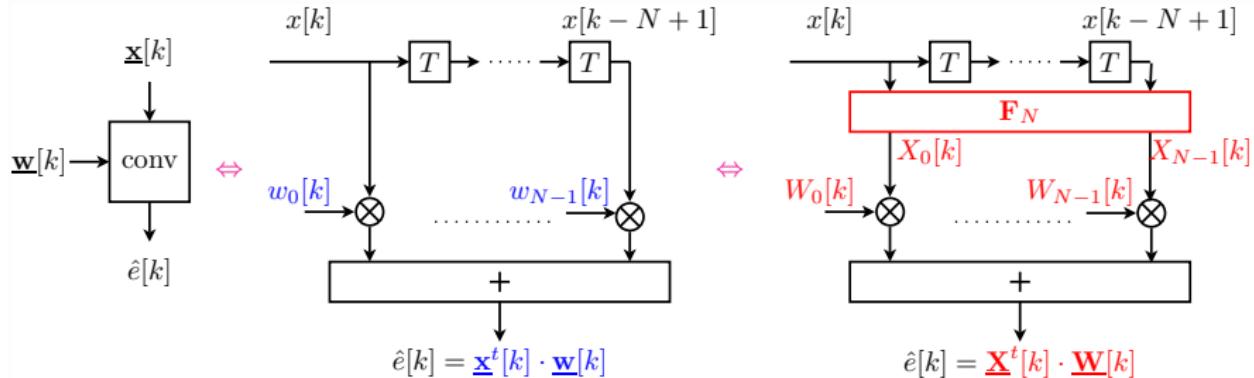
# FDAF



# FDAF



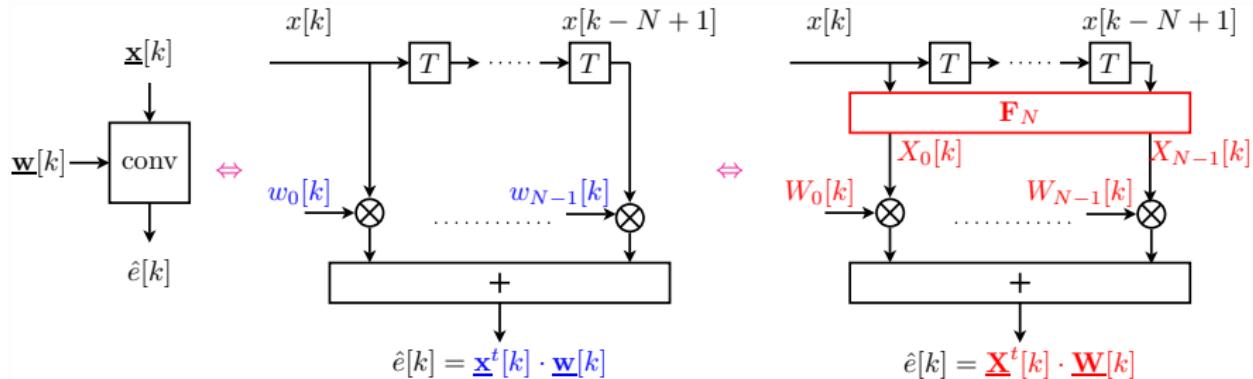
# FDAF



Notes:

- ▶ Inverse DFT in definition of weights  $W_l[k]$

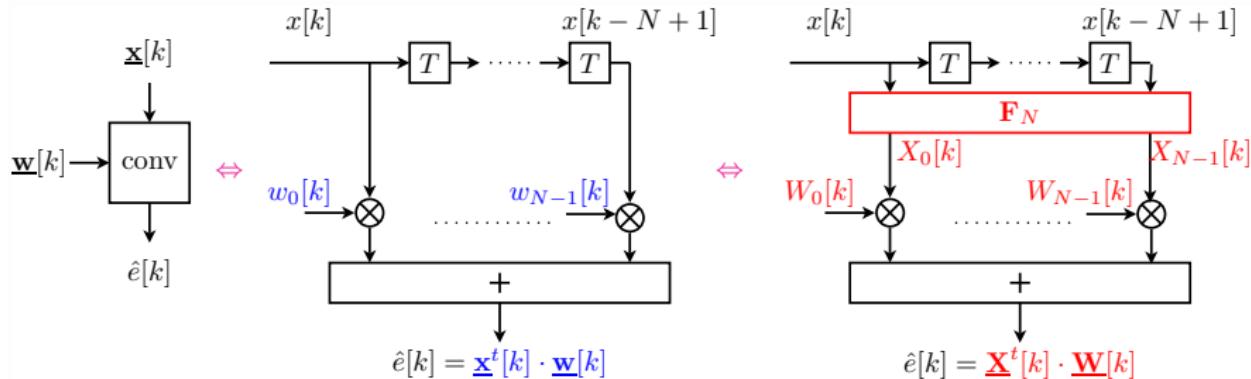
# FDAF



## Notes:

- ▶ Inverse DFT in definition of weights  $W_l[k]$
- ▶ Use DFT symmetry to reduce complexity

# FDAF



## Notes:

- ▶ Inverse DFT in definition of weights  $W_l[k]$
- ▶ Use DFT symmetry to reduce complexity
- ▶ For large  $N$ : Frequency bins "uncorrelated"

# FDAF

*Transform LMS to frequency domain (multiply update algorithm by  $\mathbf{F}^{-1}$ )*

# FDAF

*Transform LMS to frequency domain (multiply update algorithm by  $\mathbf{F}^{-1}$ )*

$$\mathbf{F}^{-1} \cdot \underline{\mathbf{w}}[k+1] = \mathbf{F}^{-1} \cdot \underline{\mathbf{w}}[k] + 2\alpha \mathbf{F}^{-1} \cdot \underline{\mathbf{x}}[k]r[k]$$

# FDAF

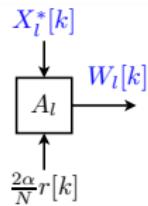
Transform LMS to frequency domain (multiply update algorithm by  $\mathbf{F}^{-1}$ )

$$\begin{aligned}\mathbf{F}^{-1} \cdot \underline{\mathbf{w}}[k+1] &= \mathbf{F}^{-1} \cdot \underline{\mathbf{w}}[k] + 2\alpha \mathbf{F}^{-1} \cdot \underline{\mathbf{x}}[k]r[k] \\ \Leftrightarrow \underline{\mathbf{W}}[k+1] &= \underline{\mathbf{W}}[k] + \frac{2\alpha}{N} \underline{\mathbf{X}}^*[k]r[k]\end{aligned}$$

# FDAF

Transform LMS to frequency domain (multiply update algorithm by  $\mathbf{F}^{-1}$ )

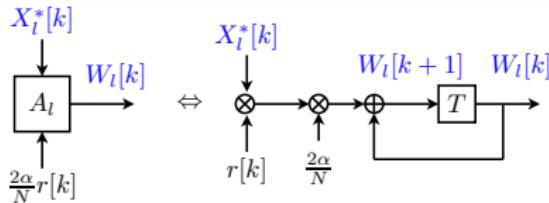
$$\begin{aligned}\mathbf{F}^{-1} \cdot \underline{\mathbf{w}}[k+1] &= \mathbf{F}^{-1} \cdot \underline{\mathbf{w}}[k] + 2\alpha \mathbf{F}^{-1} \cdot \underline{\mathbf{x}}[k]r[k] \\ \Leftrightarrow \underline{\mathbf{W}}[k+1] &= \underline{\mathbf{W}}[k] + \frac{2\alpha}{N} \underline{\mathbf{X}}^*[k]r[k]\end{aligned}$$



# FDAF

Transform LMS to frequency domain (multiply update algorithm by  $\mathbf{F}^{-1}$ )

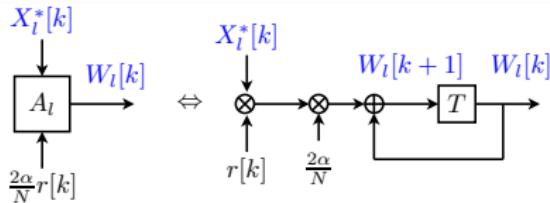
$$\begin{aligned}\mathbf{F}^{-1} \cdot \underline{\mathbf{w}}[k+1] &= \mathbf{F}^{-1} \cdot \underline{\mathbf{w}}[k] + 2\alpha \mathbf{F}^{-1} \cdot \underline{\mathbf{x}}[k]r[k] \\ \Leftrightarrow \underline{\mathbf{W}}[k+1] &= \underline{\mathbf{W}}[k] + \frac{2\alpha}{N} \underline{\mathbf{X}}^*[k]r[k]\end{aligned}$$



# FDAF

Transform LMS to frequency domain (multiply update algorithm by  $\mathbf{F}^{-1}$ )

$$\begin{aligned}\mathbf{F}^{-1} \cdot \underline{\mathbf{w}}[k+1] &= \mathbf{F}^{-1} \cdot \underline{\mathbf{w}}[k] + 2\alpha \mathbf{F}^{-1} \cdot \underline{\mathbf{x}}[k]r[k] \\ \Leftrightarrow \underline{\mathbf{W}}[k+1] &= \underline{\mathbf{W}}[k] + \frac{2\alpha}{N} \underline{\mathbf{X}}^*[k]r[k]\end{aligned}$$



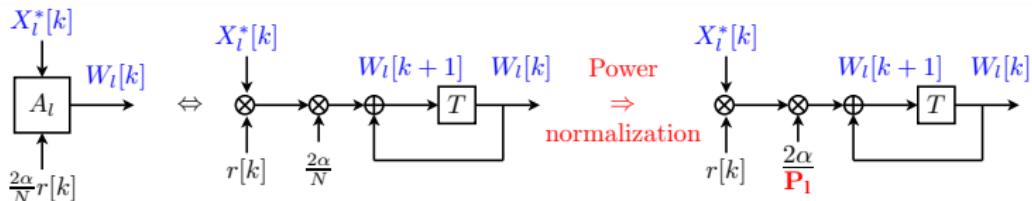
Improve convergence by Power normalization:

$$P_I = \frac{1}{N} E\{|X_l[k]|^2\}$$

# FDAF

Transform LMS to frequency domain (multiply update algorithm by  $\mathbf{F}^{-1}$ )

$$\begin{aligned}\mathbf{F}^{-1} \cdot \underline{\mathbf{w}}[k+1] &= \mathbf{F}^{-1} \cdot \underline{\mathbf{w}}[k] + 2\alpha \mathbf{F}^{-1} \cdot \underline{\mathbf{x}}[k]r[k] \\ \Leftrightarrow \underline{\mathbf{W}}[k+1] &= \underline{\mathbf{W}}[k] + \frac{2\alpha}{N} \underline{\mathbf{X}}^*[k]r[k]\end{aligned}$$



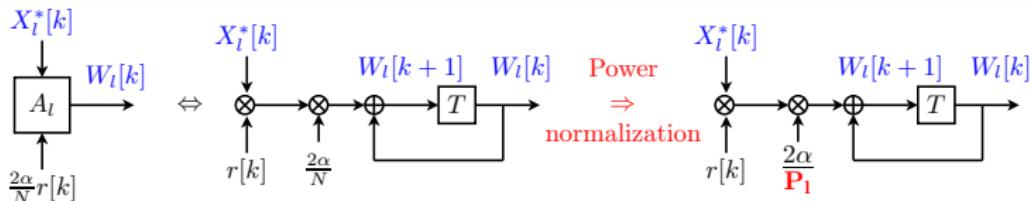
Improve convergence by Power normalization:

$$P_I = \frac{1}{N} E\{|X_l[k]|^2\}$$

# FDAF

Transform LMS to frequency domain (multiply update algorithm by  $\mathbf{F}^{-1}$ )

$$\begin{aligned}\mathbf{F}^{-1} \cdot \underline{\mathbf{w}}[k+1] &= \mathbf{F}^{-1} \cdot \underline{\mathbf{w}}[k] + 2\alpha \mathbf{F}^{-1} \cdot \underline{\mathbf{x}}[k]r[k] \\ \Leftrightarrow \underline{\mathbf{W}}[k+1] &= \underline{\mathbf{W}}[k] + \frac{2\alpha}{N} \underline{\mathbf{X}}^*[k]r[k]\end{aligned}$$



Improve convergence by Power normalization:

$$P_I = \frac{1}{N} E\{|X_l[k]|^2\} \quad \text{e.g.: } \hat{P}_l[k+1] = \beta \hat{P}_l[k] + (1 - \beta) \frac{|X_l[k]|^2}{N}$$

# FDAF

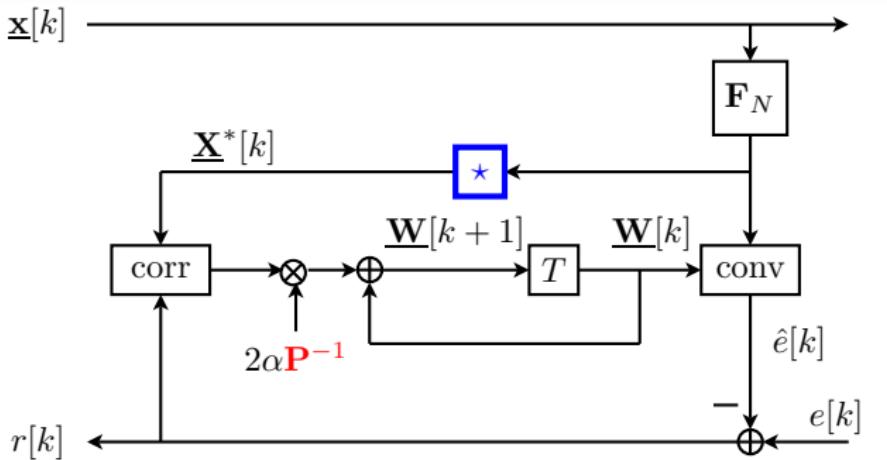
**FDAF algorithm:**  $\underline{W}[k+1] = \underline{W}[k] + 2\alpha \underline{P}^{-1} \underline{X}^*[k] r[k]$

with  $\underline{P} = \text{diag}\{\underline{P}_I\}$  and  $(\underline{P})_I = P_I = \frac{1}{N} E\{|X_I[k]|^2\}$

# FDAF

**FDAF algorithm:**  $\underline{W}[k + 1] = \underline{W}[k] + 2\alpha \underline{P}^{-1} \underline{X}^*[k] r[k]$

with  $\underline{P} = \text{diag}\{\underline{P}_I\}$  and  $(\underline{P})_I = P_I = \frac{1}{N} E\{|X_I[k]|^2\}$



# FDAF

*Average behavior FDAF:*

With  $\underline{D} = F^{-1}\underline{d} = F^{-1}(\underline{w} - \underline{w}_o) = \underline{W} - \underline{W}_o$  FDAF becomes:

# FDAF

*Average behavior FDAF:*

With  $\underline{D} = F^{-1}\underline{d} = F^{-1}(\underline{w} - \underline{w}_o) = \underline{W} - \underline{W}_o$  FDAF becomes:

$$\underline{D}[k+1] = \left( I - \frac{2\alpha}{N} P^{-1} \underline{X}^*[k] \underline{X}^t[k] \right) \underline{D}[k] + \frac{2\alpha}{N} P^{-1} \underline{X}^*[k] r_{\min}[k]$$

# FDAF

Average behavior FDAF:

With  $\underline{D} = F^{-1}\underline{d} = F^{-1}(\underline{w} - \underline{w}_o) = \underline{W} - \underline{W}_o$  FDAF becomes:

$$\underline{D}[k+1] = \left( I - \frac{2\alpha}{N} P^{-1} \underline{X}^*[k] \underline{X}^t[k] \right) \underline{D}[k] + \frac{2\alpha}{N} P^{-1} \underline{X}^*[k] r_{\min}[k]$$

Different bins 'uncorrelated'  $\Rightarrow \frac{E\{\underline{X}^*[k] \underline{X}^t[k]\}}{N} \approx P$

# FDAF

Average behavior FDAF:

With  $\underline{D} = F^{-1}\underline{d} = F^{-1}(\underline{w} - \underline{w}_o) = \underline{W} - \underline{W}_o$  FDAF becomes:

$$\underline{D}[k+1] = \left( I - \frac{2\alpha}{N} P^{-1} \underline{X}^*[k] \underline{X}^t[k] \right) \underline{D}[k] + \frac{2\alpha}{N} P^{-1} \underline{X}^*[k] r_{\min}[k]$$

Different bins 'uncorrelated'  $\Rightarrow \frac{E\{\underline{X}^*[k] \underline{X}^t[k]\}}{N} \approx P$

$$\Rightarrow E\{\underline{D}[k+1]\} \approx (1 - 2\alpha) \cdot E\{\underline{D}[k]\} \Rightarrow \lim_{k \rightarrow \infty} E\{\underline{D}[k]\} = 0$$

# FDAF

Average behavior FDAF:

With  $\underline{D} = F^{-1}\underline{d} = F^{-1}(\underline{w} - \underline{w}_o) = \underline{W} - \underline{W}_o$  FDAF becomes:

$$\underline{D}[k+1] = \left( I - \frac{2\alpha}{N} P^{-1} \underline{X}^*[k] \underline{X}^t[k] \right) \underline{D}[k] + \frac{2\alpha}{N} P^{-1} \underline{X}^*[k] r_{\min}[k]$$

Different bins 'uncorrelated'  $\Rightarrow \frac{E\{\underline{X}^*[k]\underline{X}^t[k]\}}{N} \approx P$

$$\Rightarrow E\{\underline{D}[k+1]\} \approx (1 - 2\alpha) \cdot E\{\underline{D}[k]\} \Rightarrow \lim_{k \rightarrow \infty} E\{\underline{D}[k]\} = 0$$

FDAF converges to Wiener solution:  $\lim_{k \rightarrow \infty} E\{\underline{W}[k]\} = \underline{W}_o = F^{-1}\underline{w}_o$

# FDAF

Average behavior FDAF:

With  $\underline{D} = F^{-1}\underline{d} = F^{-1}(\underline{w} - \underline{w}_o) = \underline{W} - \underline{W}_o$  FDAF becomes:

$$\underline{D}[k+1] = \left( I - \frac{2\alpha}{N} P^{-1} \underline{X}^*[k] \underline{X}^t[k] \right) \underline{D}[k] + \frac{2\alpha}{N} P^{-1} \underline{X}^*[k] r_{\min}[k]$$

Different bins 'uncorrelated'  $\Rightarrow \frac{E\{\underline{X}^*[k]\underline{X}^t[k]\}}{N} \approx P$

$$\Rightarrow E\{\underline{D}[k+1]\} \approx (1 - 2\alpha) \cdot E\{\underline{D}[k]\} \Rightarrow \lim_{k \rightarrow \infty} E\{\underline{D}[k]\} = 0$$

FDAF converges to Wiener solution:  $\lim_{k \rightarrow \infty} E\{\underline{W}[k]\} = \underline{W}_o = F^{-1}\underline{w}_o$

Notes:

- DFT (FFT) is fixed transform: Easy but not exact  
 $(\frac{E\{\underline{X}^*[k]\underline{X}^t[k]\}}{N} \approx P)$

# FDAF

Average behavior FDAF:

With  $\underline{D} = F^{-1}\underline{d} = F^{-1}(\underline{w} - \underline{w}_o) = \underline{W} - \underline{W}_o$  FDAF becomes:

$$\underline{D}[k+1] = \left( I - \frac{2\alpha}{N} P^{-1} \underline{X}^*[k] \underline{X}^t[k] \right) \underline{D}[k] + \frac{2\alpha}{N} P^{-1} \underline{X}^*[k] r_{\min}[k]$$

Different bins 'uncorrelated'  $\Rightarrow \frac{E\{\underline{X}^*[k]\underline{X}^t[k]\}}{N} \approx P$

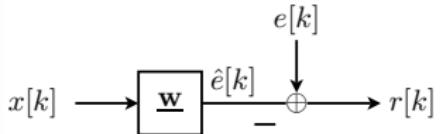
$$\Rightarrow E\{\underline{D}[k+1]\} \approx (1 - 2\alpha) \cdot E\{\underline{D}[k]\} \Rightarrow \lim_{k \rightarrow \infty} E\{\underline{D}[k]\} = 0$$

FDAF converges to Wiener solution:  $\lim_{k \rightarrow \infty} E\{\underline{W}[k]\} = \underline{W}_o = F^{-1}\underline{w}_o$

Notes:

- ▶ DFT (FFT) is fixed transform: Easy but not exact  
 $(\frac{E\{\underline{X}^*[k]\underline{X}^t[k]\}}{N} \approx P)$
- ▶ FDAF equivalent to NLMS with white noise input

# Summary



|                                 | MMSE  | LS   |
|---------------------------------|---|--|
| Auto correlation                | $R_x = E\{\underline{x}[k] \cdot \underline{x}^t[k]\}$        | $\bar{R}_x = \mathbf{X}^t \cdot \mathbf{X}$                                  |
| Cross correlation               | $\underline{r}_{ex} = E\{e[k] \cdot \underline{x}[k]\}$       | $\bar{r}_{ex} = \mathbf{X}^t \cdot \underline{e}$                            |
| Error $J$                       | $E\{r^2[k]\}$   | $\sum_{i=0}^{L-1} r^2[k-i]$  |
| Criterion                       | $\min_{\underline{w}} \{E\{r^2[k]\}\}$                        | $\min_{\underline{w}}  \underline{e} - \mathbf{X} \cdot \underline{w} ^2$    |
| Opt. solution $\underline{w}_o$ | $R_x^{-1} \cdot \underline{r}_{ex}$                           | $\bar{R}_x^{-1} \cdot \bar{r}_{ex}$  |
| Min. error $J_{min}$            | $E\{e^2\} - \underline{r}_{ex}^t R_x^{-1} \underline{r}_{ex}$ | $\underline{e}^t \underline{e} - \bar{r}_{ex}^t \bar{R}_x^{-1} \bar{r}_{ex}$ |

## Summary

Set of constraints:  $\mathbf{C}^t \cdot \underline{\mathbf{w}} = \underline{\mathbf{f}}$

Solution for  $N \geq M$ :  $\underline{\mathbf{w}}^c = \mathbf{C}(\mathbf{C}^t \mathbf{C})^{-1} \underline{\mathbf{f}}$

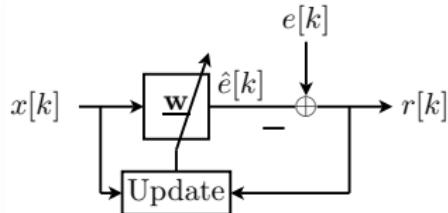
Solution for  $N > M$  with MMSE:

$$\underline{\mathbf{w}}_o^c = \underline{\mathbf{w}}_o + \mathbf{R}_x^{-1} \mathbf{C} (\mathbf{C}^t \mathbf{R}_x^{-1} \mathbf{C})^{-1} (\underline{\mathbf{f}} - \mathbf{C}^t \underline{\mathbf{w}}_o)$$

Similar result:

$$\underline{\mathbf{w}}_o^c = \mathbf{R}_x^{-1} \mathbf{C} (\mathbf{C}^t \mathbf{R}_x^{-1} \mathbf{C})^{-1} \underline{\mathbf{f}}$$

# Summary



Simple adaptive algorithms (no decorrelation):

$$\text{SGD} : \underline{w}[k+1] = \underline{w}[k] + 2\alpha(\underline{r}_{ex} - \mathbf{R}_x \underline{w}[k])$$

$$(\text{complex})(N)\text{LMS} : \underline{w}[k+1] = \underline{w}[k] + \frac{2\alpha}{\hat{\sigma}_x^2} \underline{x}[k] \underline{r}^*[k]$$

Constrained LMS:  $C^t \cdot \underline{w} = \underline{f}$

$$\underline{w}[k+1] = \tilde{\mathbf{P}} \cdot \{\underline{w}[k] + 2\alpha \underline{x}[k] \underline{r}[k]\} + C(C^t \cdot C)^{-1} \underline{f}$$

with

$$\tilde{\mathbf{P}} = \mathbf{I} - C(C^t \cdot C)^{-1} C^t \text{ and } \underline{w}[0] = C(C^t \cdot C)^{-1} \underline{f}$$

# Summary

Algorithms with improved convergence:

$$\text{LMS/Newton} : \underline{w}[k+1] = \underline{w}[k] + 2\alpha R_x^{-1} \underline{x}[k] r[k]$$

$$\text{Newton} : \underline{w}[k+1] = \underline{w}[k] + 2\alpha R_x^{-1} \cdot (\underline{r}_{ex} - R_x \underline{w}[k])$$

$$\text{RLS} : \underline{g}[k+1] = \frac{\bar{R}_x^{-1}[k] \underline{x}[k+1]}{\gamma^2 + \underline{x}^t[k+1] \bar{R}_x^{-1}[k] \underline{x}[k+1]}$$

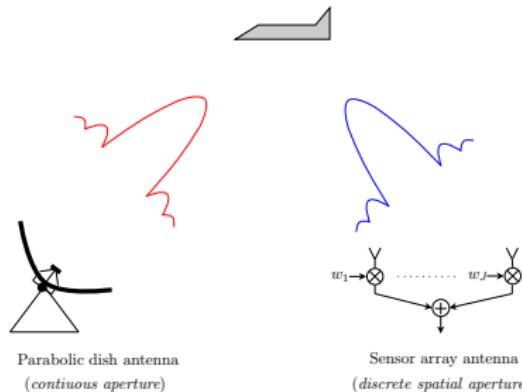
$$\bar{R}_x^{-1}[k+1] = \gamma^{-2} \left( \bar{R}_x^{-1}[k] - \underline{g}[k+1] \cdot \underline{x}^t[k+1] \bar{R}_x^{-1}[k] \right)$$

$$\bar{r}_{ex}[k+1] = \gamma^2 \bar{r}_{ex}[k] + \underline{x}^t[k+1] \cdot e[k+1]$$

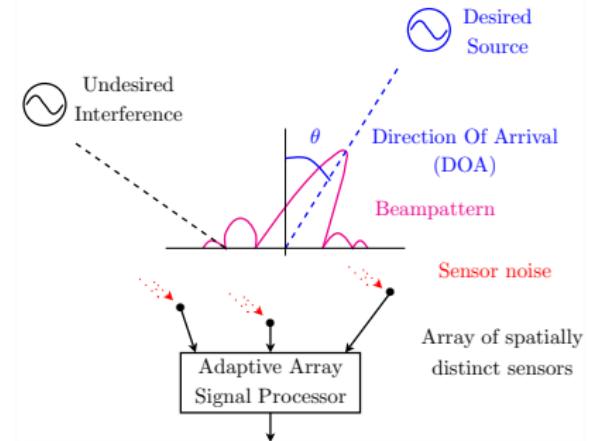
$$\underline{w}[k+1] = \bar{R}_x^{-1}[k+1] \cdot \underline{r}_{ex}[k+1]$$

# Array Signal Processing (Part IB)

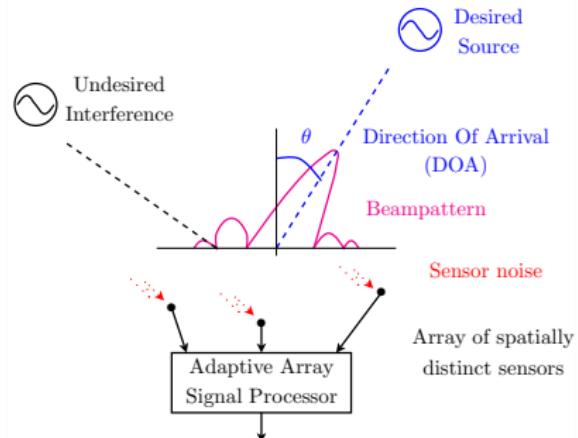
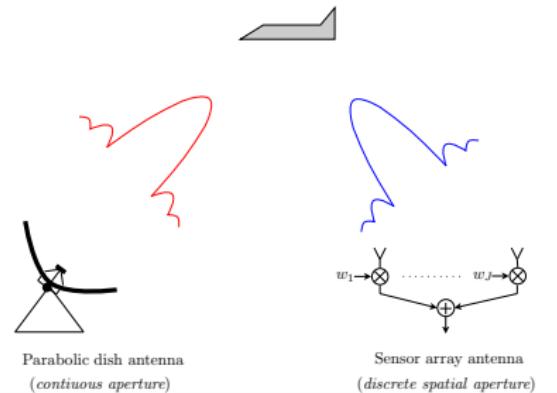
# Introduction



Parabolic dish antenna  
(continuous aperture)

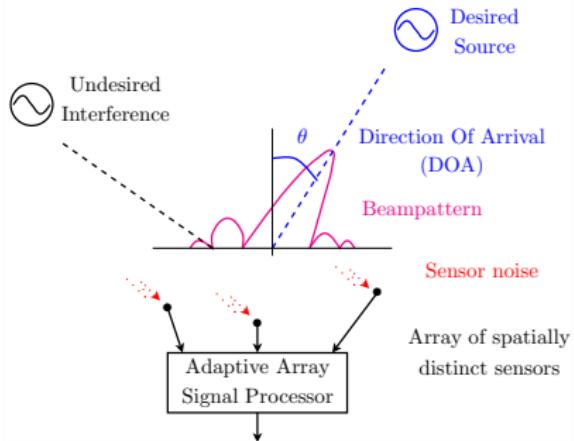
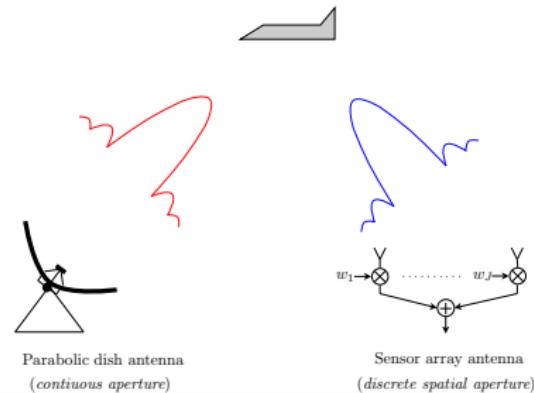


# Introduction



**Beamforming:** Spatio temporal filtering to either direct or block the radiation or reception of signals in specified directions

# Introduction



**Beamforming:** Spatio temporal filtering to either direct or block the radiation or reception of signals in specified directions

Result Beamforming = **Spatial filtering:** Separate signals with possible overlapping frequencies but from **different** directions

## **Different scenario's (content first part):**

- ▶ Bandwidth source
- ▶ Array geometry
- ▶ Far field vs. near field
- ▶ Direction Of Arrival (DOA)
- ▶ Discrete-time signal representation
- ▶ Array signal model
- ▶ ASP unit
- ▶ Spatial/ temporal filtering
- ▶ Broadband signals

## Scenario: Bandwidth source

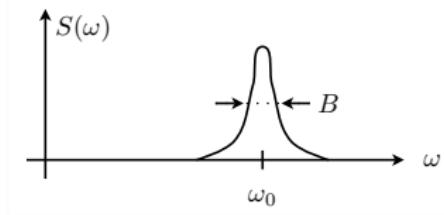
Analytical representation:  $s(t) = A(t)e^{j(\omega_o t + \phi(t))}$

## Scenario: Bandwidth source

Analytical representation:  $s(t) = A(t)e^{j(\omega_0 t + \phi(t))}$

**Narrowband:**  $A(t)$  and  $\phi(t)$  vary slower than  $e^{j\omega_0 t}$

Narrowband:  $|\tau| \ll 1/B$



$A(t - \tau) \approx A(t) = 1$  (usually)  $\Rightarrow$

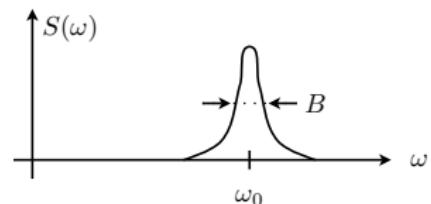
$\phi(t - \tau) \approx \phi(t) = 0$  (usually)

## Scenario: Bandwidth source

Analytical representation:  $s(t) = A(t)e^{j(\omega_0 t + \phi(t))}$

**Narrowband:**  $A(t)$  and  $\phi(t)$  vary slower than  $e^{j\omega_0 t}$

Narrowband:  $|\tau| \ll 1/B$



$A(t - \tau) \approx A(t) = 1$  (usually)  $\Rightarrow$

$\phi(t - \tau) \approx \phi(t) = 0$  (usually)

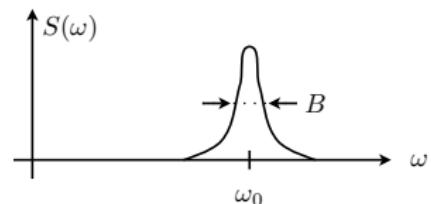
$$\Rightarrow s(t - \tau) = A(t - \tau)e^{j\phi(t - \tau)}e^{j\omega_0(t - \tau)} \approx e^{-j\omega_0\tau} \cdot s(t)$$

## Scenario: Bandwidth source

Analytical representation:  $s(t) = A(t)e^{j(\omega_0 t + \phi(t))}$

**Narrowband:**  $A(t)$  and  $\phi(t)$  vary slower than  $e^{j\omega_0 t}$

Narrowband:  $|\tau| \ll 1/B$



$A(t - \tau) \approx A(t) = 1$  (usually)  $\Rightarrow$

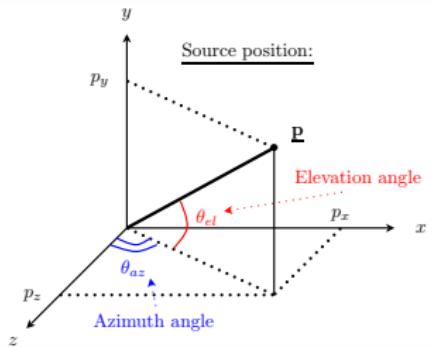
$\phi(t - \tau) \approx \phi(t) = 0$  (usually)

$$\Rightarrow s(t - \tau) = A(t - \tau)e^{j\phi(t - \tau)}e^{j\omega_0(t - \tau)} \approx e^{-j\omega_0\tau} \cdot s(t)$$

Thus for narrowband: Time delay  $\equiv$  phase shift

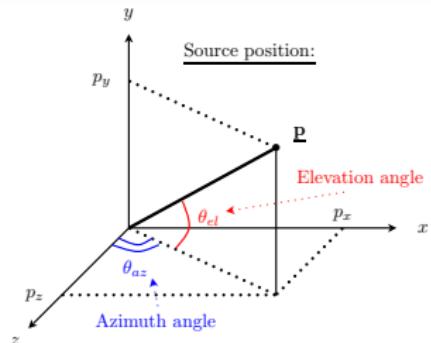
*In this course mainly narrowband*

## Scenario: Array geometry



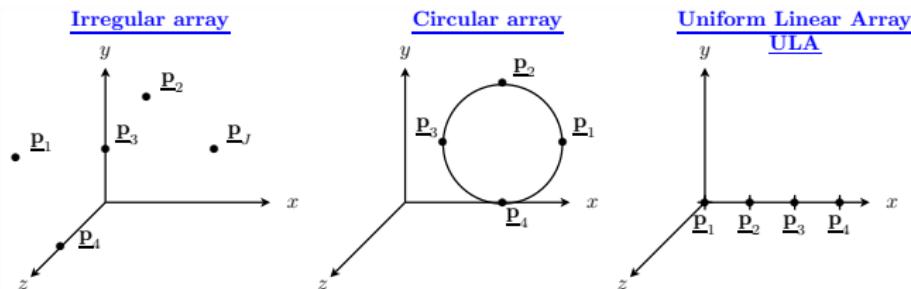
$$\begin{aligned}\underline{p} &= (p_x, p_y, p_z)^t \\ p_x &= \|\underline{p}\| \sin(\theta_{az}) \cos(\theta_{el}) \\ p_y &= \|\underline{p}\| \sin(\theta_{el}) \\ p_z &= \|\underline{p}\| \cos(\theta_{az}) \cos(\theta_{el})\end{aligned}$$

# Scenario: Array geometry



$$\begin{aligned}\underline{p} &= (p_x, p_y, p_z)^t \\ p_x &= \|\underline{p}\| \sin(\theta_{az}) \cos(\theta_{el}) \\ p_y &= \|\underline{p}\| \sin(\theta_{el}) \\ p_z &= \|\underline{p}\| \cos(\theta_{az}) \cos(\theta_{el})\end{aligned}$$

Array can be uniform, nonuniform, linear, circular, · · ·



*In this course mainly: ULA*

# Scenario: Far field vs. near field

**Array aperture:** Volume (1D length) that collects incoming signal



Near field

- Distance source - array  $\ll$  Array aperture
- Spherical wavefront

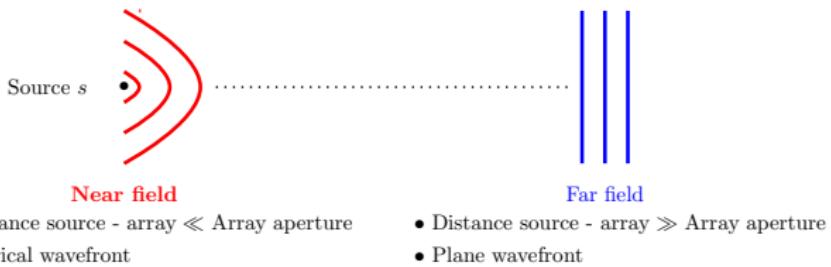


Far field

- Distance source - array  $\gg$  Array aperture
- Plane wavefront

# Scenario: Far field vs. near field

**Array aperture:** Volume (1D length) that collects incoming signal



**Near field:** Propagation for single frequency source

$$s(t, \underline{p}) = \frac{A}{\|\underline{p}\|^2} e^{j\omega(t - \frac{\|\underline{p}\|}{c})} \text{ with } \omega = 2\pi f \text{ and } f = \frac{c}{\lambda}$$

$\lambda$  = wavelength,  $c$  = speed in medium ( $\approx 334$  [m/sec] for sound in air)

⇒ Amplitude decays proportional to distance from source

## Scenario: Far field vs. near field

*In this course mainly far field:*

$$s(t, \underline{p}_i) = A e^{j\omega(t-\tau_i)} = A e^{j\omega(t - \frac{\underline{v}^t \cdot \underline{p}_i}{c})} = A e^{j(\omega t - \underline{k}^t \cdot \underline{p}_i)}$$

with direction vector  $\underline{v}$ , wave number vector  $\underline{k} = \frac{\omega}{c} \cdot \underline{v}$

## Scenario: Far field vs. near field

*In this course mainly far field:*

$$s(t, \underline{p}_i) = A e^{j\omega(t-\tau_i)} = A e^{j\omega(t - \frac{\underline{v}^t \cdot \underline{p}_i}{c})} = A e^{j(\omega t - \underline{k}^t \cdot \underline{p}_i)}$$

with direction vector  $\underline{v}$ , wave number vector  $\underline{k} = \frac{\omega}{c} \cdot \underline{v}$

- ✓  $s(t, \underline{p}_i)$  describes propagation as function of both time and space
- ✓ Information is preserved while propagating

## Scenario: Far field vs. near field

*In this course mainly far field:*

$$s(t, \underline{p}_i) = A e^{j\omega(t-\tau_i)} = A e^{j\omega(t - \frac{\underline{v}^t \cdot \underline{p}_i}{c})} = A e^{j(\omega t - \underline{k}^t \cdot \underline{p}_i)}$$

with direction vector  $\underline{v}$ , wave number vector  $\underline{k} = \frac{\omega}{c} \cdot \underline{v}$

- ✓  $s(t, \underline{p}_i)$  describes propagation as function of both time and space
  - ✓ Information is preserved while propagating
- ⇒ Reconstruction band limited signal over all space and time by either:
- ▶ Temporally sampling at given location in space
  - ▶ Spatially sampling at given instant of time
  - ▶ Combination

## Scenario: Far field vs. near field

*In this course mainly far field:*

$$s(t, \underline{p}_i) = A e^{j\omega(t-\tau_i)} = A e^{j\omega(t - \frac{\underline{v}^t \cdot \underline{p}_i}{c})} = A e^{j(\omega t - \underline{k}^t \cdot \underline{p}_i)}$$

with direction vector  $\underline{v}$ , wave number vector  $\underline{k} = \frac{\omega}{c} \cdot \underline{v}$

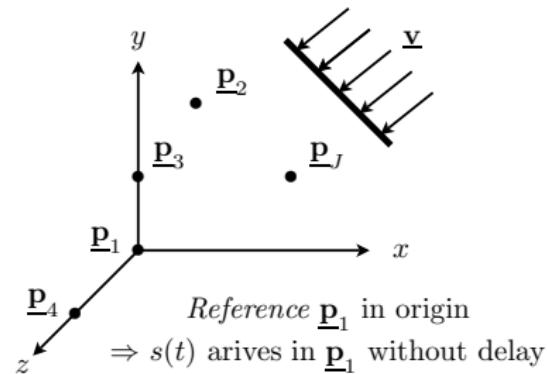
- ✓  $s(t, \underline{p}_i)$  describes propagation as function of both time and space
  - ✓ Information is preserved while propagating
- ⇒ Reconstruction band limited signal over all space and time by either:

- ▶ Temporally sampling at given location in space
- ▶ Spatially sampling at given instant of time
- ▶ Combination

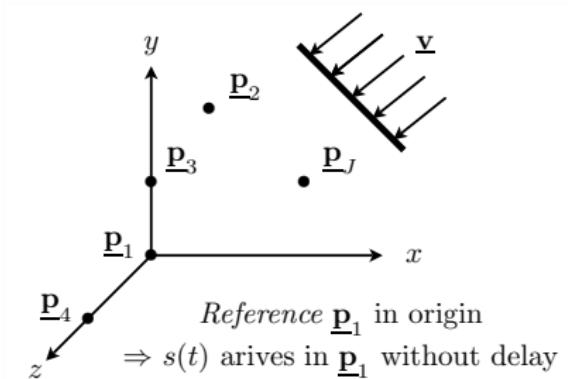
Spatially sampling:

Basis for all aperture and sensor array processing techniques

## Scenario: Far field vs. near field



## Scenario: Far field vs. near field



At position  $\underline{p}_i$ :  $s(t - \tau_i) = s(t)e^{-j\omega\tau_i}$

with delay  $\tau_i = \frac{\underline{v}^t \cdot \underline{p}_i}{c}$  and  $\underline{v}$  is direction vector

$\omega = 2\pi f$ ,  $f = \frac{c}{\lambda}$ ,  $\lambda$  = wavelength,  $c$  = speed in medium

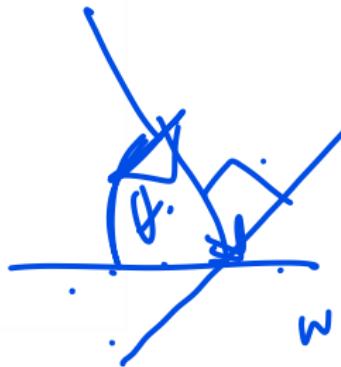
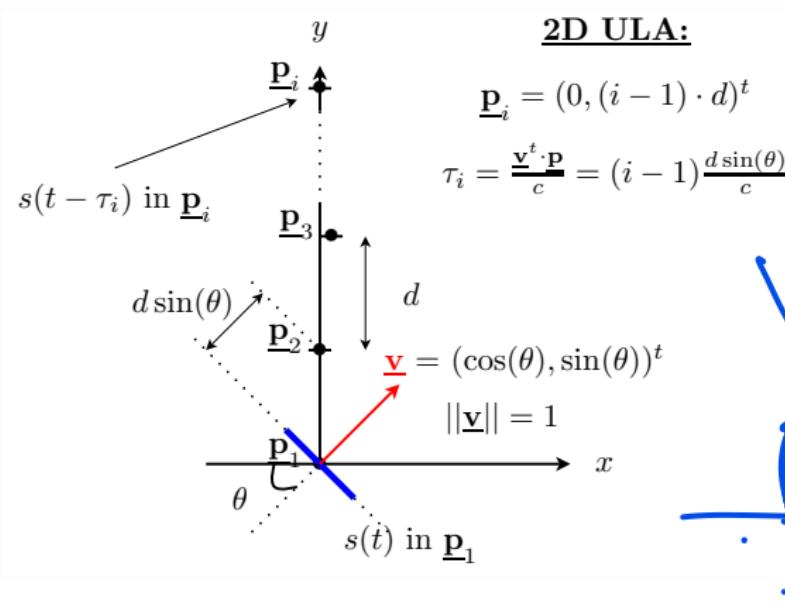
## Scenario: Direction Of Arrival (DOA)

Location is 3D quantity. In practice often 2D

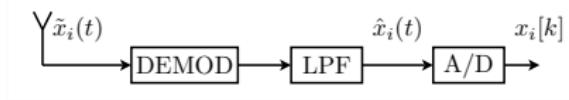
# Scenario: Direction Of Arrival (DOA)

Location is 3D quantity. In practice often 2D

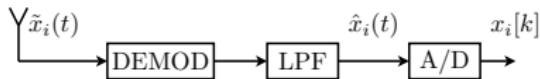
Example: Narrow band, far field 2D DOA for ULA:



## Scenario: Discrete-time signal representation



## Scenario: Discrete-time signal representation



- ✓ Analog sensor signal at sensor  $i$ :  $\tilde{x}_i(t)$
- ✓ Ideal demodulation and LPF results in baseband signal:  $\hat{x}_i(t)$
- ✓ After A/D (complex valued) discrete-time signal:  $x_i[k]$
- ✓ Analog signal at  $\underline{p}_i$  for narrow band, far field case:

$$\hat{x}_i(t) = s(t - \tau_i) = s(t)e^{-j\omega\tau_i} \quad \text{with} \quad \tau_i = \frac{\underline{v}^t \cdot \underline{p}_i}{c}$$

## Scenario: Discrete-time signal representation



- ✓ Analog sensor signal at sensor  $i$ :  $\tilde{x}_i(t)$
- ✓ Ideal demodulation and LPF results in baseband signal:  $\hat{x}_i(t)$
- ✓ After A/D (complex valued) discrete-time signal:  $x_i[k]$
- ✓ Analog signal at  $\underline{p}_i$  for narrow band, far field case:

$$\hat{x}_i(t) = s(t - \tau_i) = s(t)e^{-j\omega\tau_i} \quad \text{with} \quad \tau_i = \frac{\underline{v}^t \cdot \underline{p}_i}{c}$$

- ✓ Discrete-time signal at  $\underline{p}_i$  for ULA-case:

$$s[k]e^{-j\omega\tau_i} = s[k] \cdot e^{-j2\pi(i-1)\frac{d \sin(\theta)}{\lambda}} = s[k] \cdot a_i(\theta) \quad \text{with } a_i(\theta) = e^{-j2\pi(i-1)\frac{d \sin(\theta)}{\lambda}}$$

## Scenario: Discrete-time signal representation



- ✓ Analog sensor signal at sensor  $i$ :  $\tilde{x}_i(t)$
- ✓ Ideal demodulation and LPF results in baseband signal:  $\hat{x}_i(t)$
- ✓ After A/D (complex valued) discrete-time signal:  $x_i[k]$
- ✓ Analog signal at  $\underline{p}_i$  for narrow band, far field case:

$$\hat{x}_i(t) = s(t - \tau_i) = s(t)e^{-j\omega\tau_i} \quad \text{with} \quad \tau_i = \frac{\underline{v}^t \cdot \underline{p}_i}{c}$$

- ✓ Discrete-time signal at  $\underline{p}_i$  for ULA-case:

$$s[k]e^{-j\omega\tau_i} = s[k] \cdot e^{-j2\pi(i-1)\frac{d \sin(\theta)}{\lambda}} = s[k] \cdot a_i(\theta) \quad \text{with } a_i(\theta) = e^{-j2\pi(i-1)\frac{d \sin(\theta)}{\lambda}}$$

Note: In fact  $a_i(\theta)$  also depends on  $\omega$

## Scenario: Array signal model

Array sensor vector :  $\underline{x}[k] = (x_1[k], x_2[k], \dots, x_J[k])^t$

Noise vector :  $\underline{n}[k] = (n_1[k], n_2[k], \dots, n_J[k])^t$

Steering vector :  $\underline{a}[k] = (a_1(\theta), a_2(\theta), \dots, a_J(\theta))^t$   
with  $a_i(\theta) = e^{-j\omega\tau_i(\theta)}$

## Scenario: Array signal model

Array sensor vector :  $\underline{x}[k] = (x_1[k], x_2[k], \dots, x_J[k])^t$

Noise vector :  $\underline{n}[k] = (n_1[k], n_2[k], \dots, n_J[k])^t$

Steering vector :  $\underline{a}[k] = (a_1(\theta), a_2(\theta), \dots, a_J(\theta))^t$   
with  $a_i(\theta) = e^{-j\omega\tau_i(\theta)}$

Case: Noise observation,  $P$  sources,  $J$  sensors

## Scenario: Array signal model

Array sensor vector :  $\underline{x}[k] = (x_1[k], x_2[k], \dots, x_J[k])^t$

Noise vector :  $\underline{n}[k] = (n_1[k], n_2[k], \dots, n_J[k])^t$

Steering vector :  $\underline{a}[k] = (a_1(\theta), a_2(\theta), \dots, a_J(\theta))^t$   
with  $a_i(\theta) = e^{-j\omega\tau_i(\theta)}$

### Case: Noise observation, $P$ sources, $J$ sensors

$$x_i[k] = \sum_{p=1}^P a_i(\theta_p) s_p[k] + n_i[k]$$

## Scenario: Array signal model

Array sensor vector :  $\underline{x}[k] = (x_1[k], x_2[k], \dots, x_J[k])^t$

Noise vector :  $\underline{n}[k] = (n_1[k], n_2[k], \dots, n_J[k])^t$

Steering vector :  $\underline{a}[k] = (a_1(\theta), a_2(\theta), \dots, a_J(\theta))^t$   
with  $a_i(\theta) = e^{-j\omega\tau_i(\theta)}$

### Case: Noise observation, $P$ sources, $J$ sensors

$$x_i[k] = \sum_{p=1}^P a_i(\theta_p) s_p[k] + n_i[k] \Leftrightarrow \boxed{\underline{x}[k] = \mathbf{A} \cdot \underline{s}[k] + \underline{n}[k]}$$

## Scenario: Array signal model

Array sensor vector :  $\underline{x}[k] = (x_1[k], x_2[k], \dots, x_J[k])^t$

Noise vector :  $\underline{n}[k] = (n_1[k], n_2[k], \dots, n_J[k])^t$

Steering vector :  $\underline{a}[k] = (a_1(\theta), a_2(\theta), \dots, a_J(\theta))^t$   
with  $a_i(\theta) = e^{-j\omega\tau_i(\theta)}$

### Case: Noise observation, $P$ sources, $J$ sensors

$$x_i[k] = \sum_{p=1}^P a_i(\theta_p) s_p[k] + n_i[k] \Leftrightarrow \boxed{\underline{x}[k] = \mathbf{A} \cdot \underline{s}[k] + \underline{n}[k]}$$

$J \times P$  steering matrix  $\mathbf{A} = (\underline{a}(\theta_1), \underline{a}(\theta_2), \dots, \underline{a}(\theta_P))$

$P \times 1$  signal vector  $\underline{s}[k] = (s_1[k], s_2[k], \dots, s_P[k])^t$

## Scenario: Array signal model

Array sensor vector :  $\underline{x}[k] = (x_1[k], x_2[k], \dots, x_J[k])^t$

Noise vector :  $\underline{n}[k] = (n_1[k], n_2[k], \dots, n_J[k])^t$

Steering vector :  $\underline{a}[k] = (a_1(\theta), a_2(\theta), \dots, a_J(\theta))^t$   
with  $a_i(\theta) = e^{-j\omega\tau_i(\theta)}$

### Case: Noise observation, $P$ sources, $J$ sensors

$$x_i[k] = \sum_{p=1}^P a_i(\theta_p) s_p[k] + n_i[k] \Leftrightarrow \boxed{\underline{x}[k] = \mathbf{A} \cdot \underline{s}[k] + \underline{n}[k]}$$

$J \times P$  steering matrix  $\mathbf{A} = (\underline{a}(\theta_1), \underline{a}(\theta_2), \dots, \underline{a}(\theta_P))$

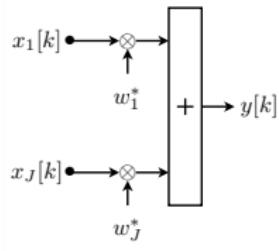
$P \times 1$  signal vector  $\underline{s}[k] = (s_1[k], s_2[k], \dots, s_P[k])^t$

Covariance structure:  $\boxed{\mathbf{R}_x = E\{\underline{x} \cdot \underline{x}^h\} = \mathbf{A} \mathbf{R}_s \mathbf{A}^h + \mathbf{R}_n}$

with  $\mathbf{R}_s = E\{\underline{s} \cdot \underline{s}^h\}$  and  $\mathbf{R}_n = E\{\underline{n} \cdot \underline{n}^h\} = \sigma_n^2 \mathbf{I}$

# Scenario: ASP unit

Case: Single complex weight for each sensor



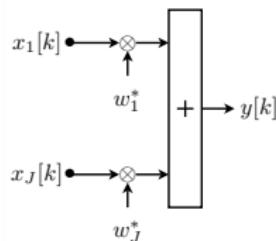
$$y[k] = \sum_{i=1}^J w_i^* x_i[k] = \underline{w}^h \cdot \underline{x}[k]$$

$$\underline{x}[k] = (x_1[k], \dots, x_J[k])^t$$

$$\underline{w} = (w_1, \dots, w_J)^t$$

# Scenario: ASP unit

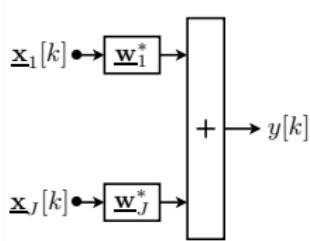
Case: Single complex weight for each sensor



$$y[k] = \sum_{i=1}^J w_i^* x_i[k] = \underline{w}^h \cdot \underline{x}[k]$$

$$\begin{aligned}\underline{x}[k] &= (x_1[k], \dots, x_J[k])^t \\ \underline{w} &= (w_1, \dots, w_J)^t\end{aligned}$$

Case: FIR filter for each sensor

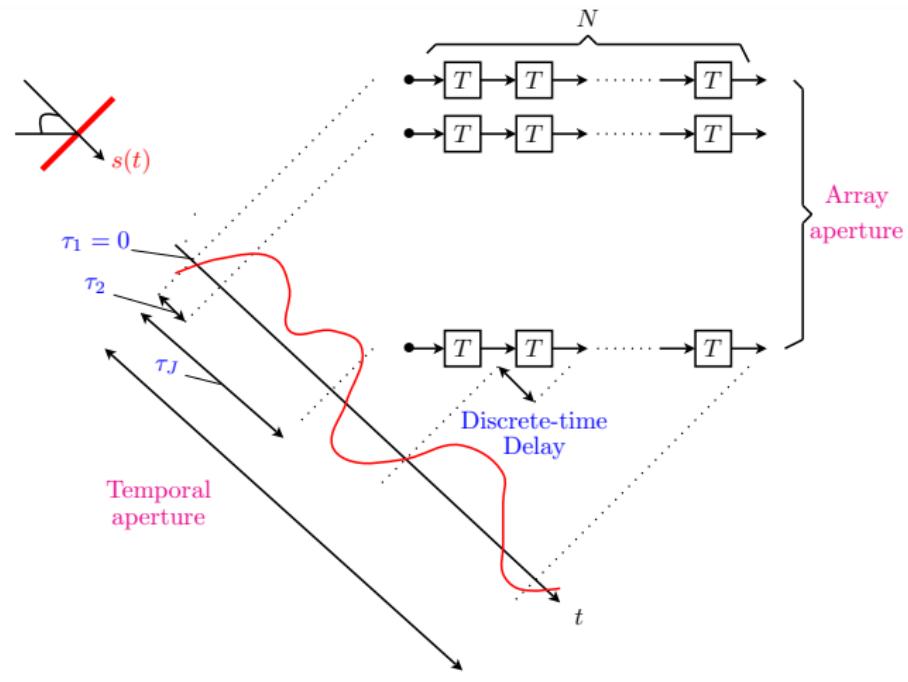


$\underline{w}_i$ : FIR filter with  $N$  weights

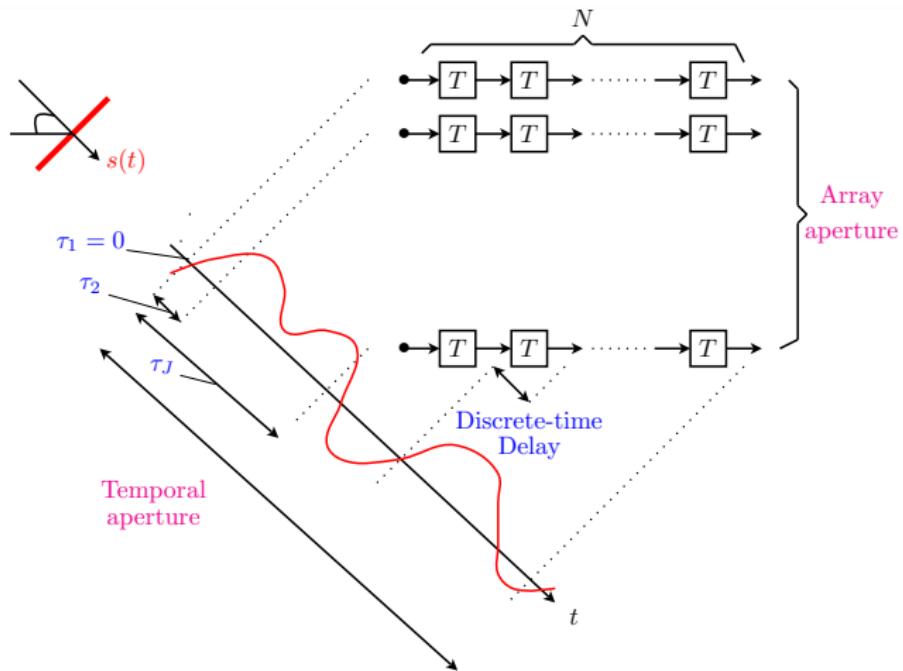
$$y[k] = \sum_{i=1}^J \underline{w}_i^h \cdot \underline{x}_i[k] = \underline{w}^h \cdot \underline{x}[k]$$

$$\begin{aligned}\underline{x}[k] &= (\underline{x}_1[k], \dots, \underline{x}_J[k])^t \\ \underline{w} &= (\underline{w}_1, \dots, \underline{w}_J)^t \\ \underline{w}_i &= (w_{i,1}, \dots, w_{i,N})^t\end{aligned}$$

# Scenario: Spatial/ temporal filtering

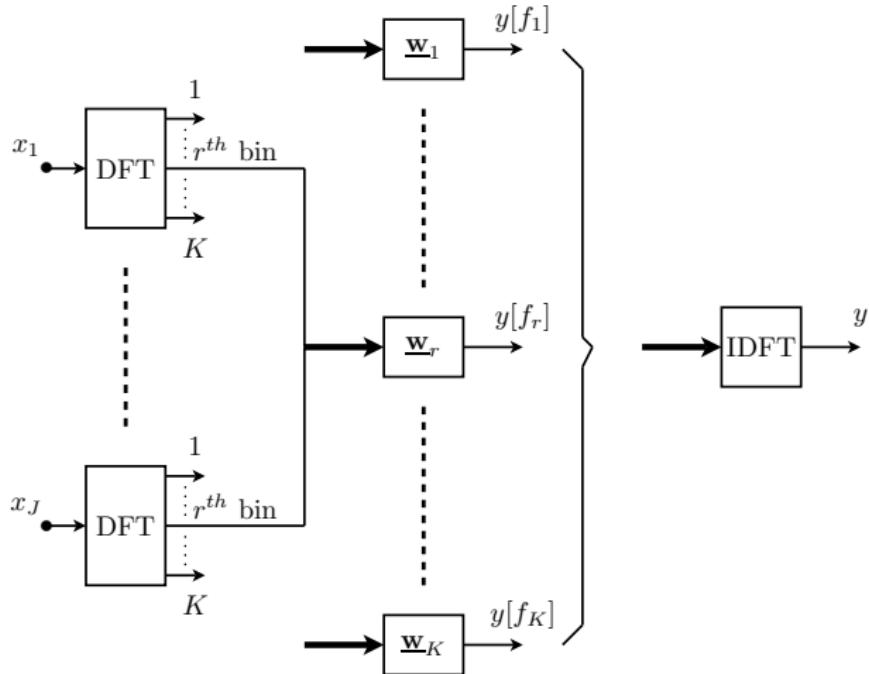


## Scenario: Spatial/ temporal filtering



Note: FIR filtering effect **both** temporal and spatial response

## Scenario: Broadband signals



# ULA Beampattern

*Main properties*

# ULA Beampattern

## Assumptions:

# ULA Beampattern

## Assumptions:

- ▶ Single source  $s(t) = e^{j\omega t}$
- ▶ Frequency relations:  $\omega = 2\pi f = 2\pi \frac{c}{\lambda}$ , with wavenumber  $\lambda$  and speed of propagation  $c$  ( $\approx 343$  [m/sec])
- ▶ Direction Of Arrival (DOA):  $\theta$
- ▶ Far field, thus plane wavefront
- ▶ ULA with distance  $d$  [m] between sensors
- ▶  $J$  omnidirectional sensors  $\Rightarrow$  array aperture  $= J \cdot d$  [m]

# ULA Beampattern

## Assumptions:

- ▶ Single source  $s(t) = e^{j\omega t}$
- ▶ Frequency relations:  $\omega = 2\pi f = 2\pi \frac{c}{\lambda}$ , with wavenumber  $\lambda$  and speed of propagation  $c$  ( $\approx 343$  [m/sec])
- ▶ Direction Of Arrival (DOA):  $\theta$
- ▶ Far field, thus plane wavefront
- ▶ ULA with distance  $d$  [m] between sensors
- ▶  $J$  omnidirectional sensors  $\Rightarrow$  array aperture  $= J \cdot d$  [m]
- ▶ Model:  $\underline{x}[k] = \underline{a}(\theta) \cdot s[k]$  (No noise, no interference)
- ▶ ASP unit: Single complex weight  $f w_i$  or each sensor

$$\Rightarrow y[k] = \sum_{i=1}^J w_i^* x_i[k] = \underline{w}^h \cdot \underline{x}[k] = \underline{w}^h \cdot \underline{a}(\theta) \cdot s[k]$$

$$\text{with } (\underline{a}(\theta))_i = e^{-j2\pi(i-1)\frac{d \sin(\theta)}{\lambda}}$$

# ULA Beampattern

**Array response** :  $r(\theta) = \underline{w}^h \cdot \underline{a}(\theta)$

*Other names:* Angular response or directivity pattern

# ULA Beampattern

**Array response** :  $r(\theta) = \underline{w}^h \cdot \underline{a}(\theta)$

*Other names:* Angular response or directivity pattern

**Beampattern** :  $B(\theta) = \frac{1}{J^2} |r(\theta)|^2 = \frac{1}{J^2} |\underline{w}^h \cdot \underline{a}(\theta)|^2$

# ULA Beampattern

**Array response** :  $r(\theta) = \underline{w}^h \cdot \underline{a}(\theta)$

*Other names:* Angular response or directivity pattern

**Beampattern** :  $B(\theta) = \frac{1}{J^2} |r(\theta)|^2 = \frac{1}{J^2} |\underline{w}^h \cdot \underline{a}(\theta)|^2$

Notes:

- ▶ Array response: Response to unit-amplitude plane wave front from direction  $\theta$
- ▶ Non ideal sensor characteristics can be incorporated
- ▶ Weights effect both temporal and spatial response
- ▶ Vector space interpretation: Angle between  $\underline{w}$  and  $\underline{a}$
- ▶ To evaluate beampattern: Choose all weight equal to one, thus  $\underline{w} = (1, 1, \dots, 1)^t$

## ULA Beampattern: The equation

$$B(\theta) = \frac{1}{J^2} |\underline{1}^t \cdot \underline{a}(\theta)|^2$$

## ULA Beampattern: The equation

$$B(\theta) = \frac{1}{J^2} |\underline{1}^t \cdot \underline{a}(\theta)|^2 = \frac{1}{J^2} \left| \sum_{i=1}^J e^{-j2\pi(i-1)\frac{d}{\lambda} \sin(\theta)} \right|^2$$

## ULA Beampattern: The equation

$$\begin{aligned}B(\theta) &= \frac{1}{J^2} |\underline{1}^t \cdot \underline{a}(\theta)|^2 = \frac{1}{J^2} \left| \sum_{i=1}^J e^{-j2\pi(i-1)\frac{d}{\lambda} \sin(\theta)} \right|^2 \\&= \frac{1}{J^2} \left| \frac{1 - e^{-jJ2\pi\frac{d}{\lambda} \sin(\theta)}}{1 - e^{-j2\pi\frac{d}{\lambda} \sin(\theta)}} \right|^2\end{aligned}$$

## ULA Beampattern: The equation

$$\begin{aligned}B(\theta) &= \frac{1}{J^2} |\underline{1}^t \cdot \underline{a}(\theta)|^2 = \frac{1}{J^2} \left| \sum_{i=1}^J e^{-j2\pi(i-1)\frac{d}{\lambda} \sin(\theta)} \right|^2 \\&= \frac{1}{J^2} \left| \frac{1 - e^{-jJ2\pi\frac{d}{\lambda} \sin(\theta)}}{1 - e^{-j2\pi\frac{d}{\lambda} \sin(\theta)}} \right|^2 = \boxed{\frac{1}{J^2} \left| \frac{\sin(J\pi\frac{d}{\lambda} \sin(\theta))}{\sin(\pi\frac{d}{\lambda} \sin(\theta))} \right|^2}\end{aligned}$$

## ULA Beampattern: The equation

$$\begin{aligned}B(\theta) &= \frac{1}{J^2} |\underline{1}^t \cdot \underline{a}(\theta)|^2 = \frac{1}{J^2} \left| \sum_{i=1}^J e^{-j2\pi(i-1)\frac{d}{\lambda} \sin(\theta)} \right|^2 \\&= \frac{1}{J^2} \left| \frac{1 - e^{-jJ2\pi\frac{d}{\lambda} \sin(\theta)}}{1 - e^{-j2\pi\frac{d}{\lambda} \sin(\theta)}} \right|^2 = \boxed{\frac{1}{J^2} \left| \frac{\sin(J\pi\frac{d}{\lambda} \sin(\theta))}{\sin(\pi\frac{d}{\lambda} \sin(\theta))} \right|^2}\end{aligned}$$

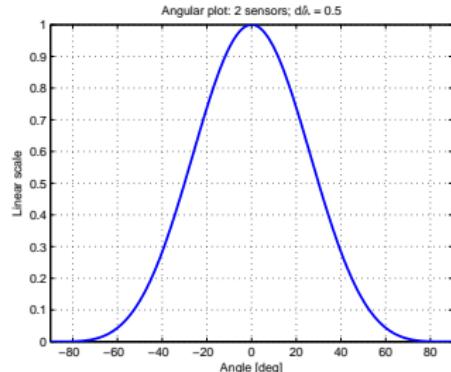
### Main parameters:

- ▶ DOA  $\theta$
- ▶ Ratio  $\frac{d}{\lambda}$  (everything scales with wavelength)
- ▶ Number of sensors  $J$
- ▶ Element spacing  $d$
- ▶ Array aperture  $L = J \cdot d$

# ULA Beampattern: Example $J = 2$ , $\frac{d}{\lambda} = \frac{1}{2}$

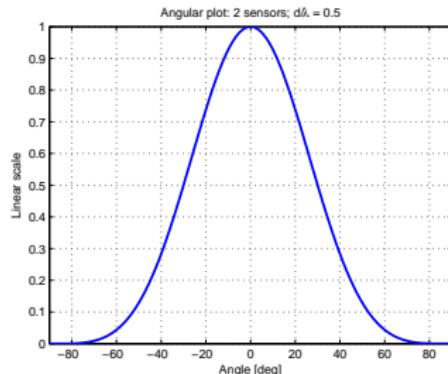
Linear plot:

Polar plot:

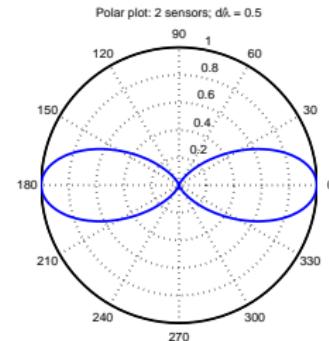


# ULA Beampattern: Example $J = 2$ , $\frac{d}{\lambda} = \frac{1}{2}$

Linear plot:

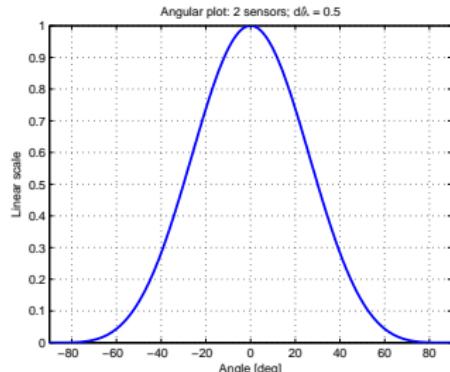


Polar plot:

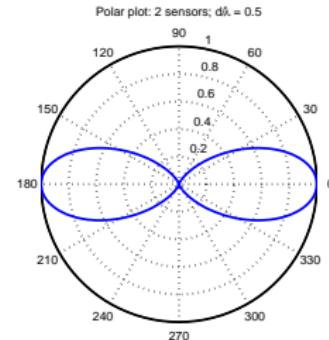


# ULA Beampattern: Example $J = 2$ , $\frac{d}{\lambda} = \frac{1}{2}$

Linear plot:



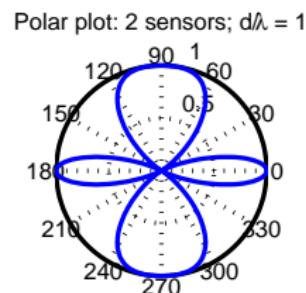
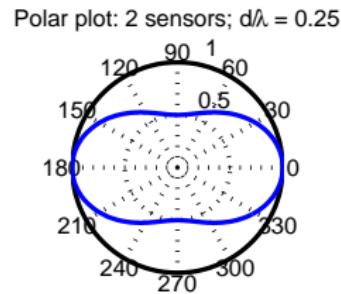
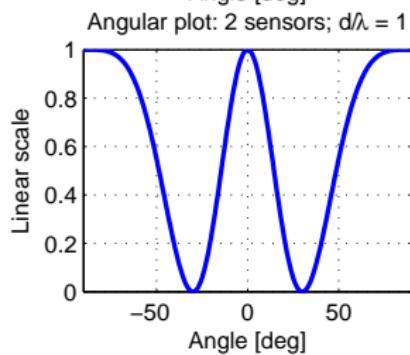
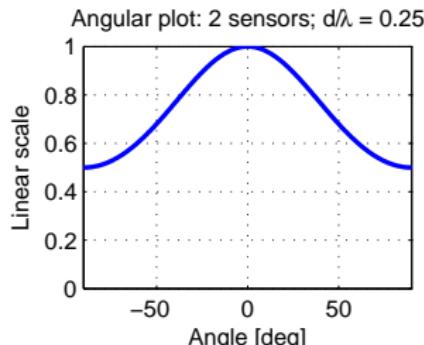
Polar plot:



## Some preliminary conclusions:

- ▶ Ambiguity between 'front' (line of sight) and 'back':  
 $B(\theta) = B(\pi - \theta)$
- ▶ Zeroes if numerator of  $B(\theta) = 0 \Rightarrow \theta = \arcsin(i \cdot \frac{1}{J} \cdot \frac{\lambda}{d})$
- ▶ For  $\frac{d}{\lambda} = \frac{1}{2} \Rightarrow$  Zeroes at  $\theta = \pm \frac{\pi}{2}$  and mainlobe (3dB)  
beamwidth:  $60^\circ$

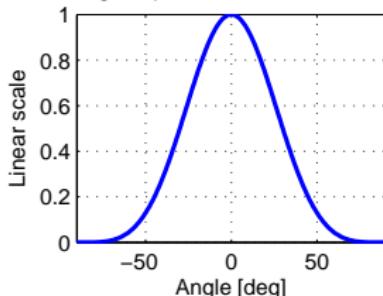
# ULA Beampattern: $J = 2$ , variable $\frac{d}{\lambda}$



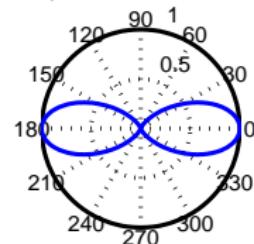
# ULA Beampattern: Variable # sensors $J$

$$L = J \cdot d \text{ with } J \uparrow \text{ and fixed } \frac{d}{\lambda} = \frac{1}{2}$$

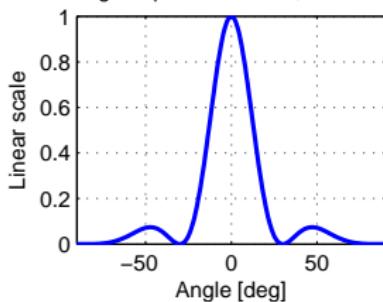
Angular plot: 2 sensors;  $d/\lambda = 0.5$



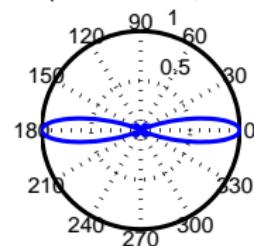
Polar plot: 2 sensors;  $d/\lambda = 0.5$



Angular plot: 4 sensors;  $d/\lambda = 0.5$



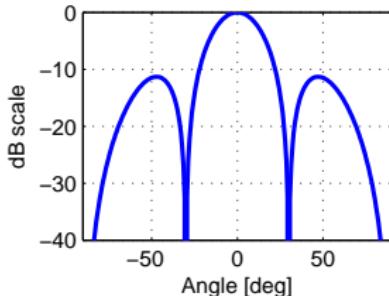
Polar plot: 4 sensors;  $d/\lambda = 0.5$



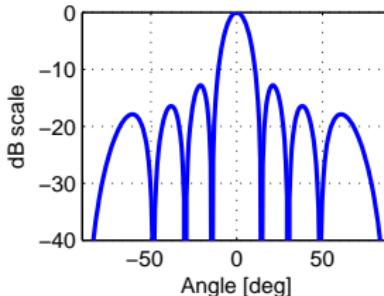
# ULA Beampattern: dB scale, variable $L$ , $\frac{d}{\lambda} = \frac{1}{2}$

$$\text{Aperture size } L = J \cdot d$$

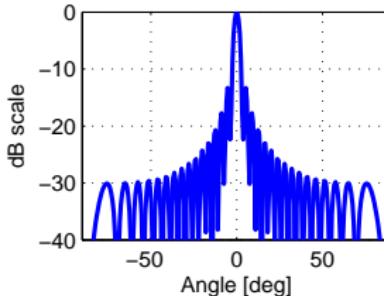
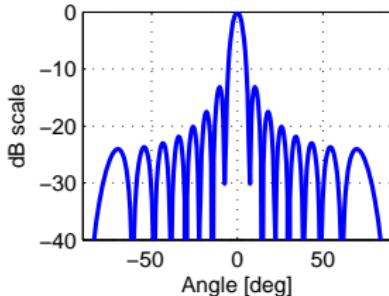
Angular plot: 4 sensors; Aperture  $L = 2\lambda$



Angular plot: 8 sensors; Aperture  $L = 4\lambda$



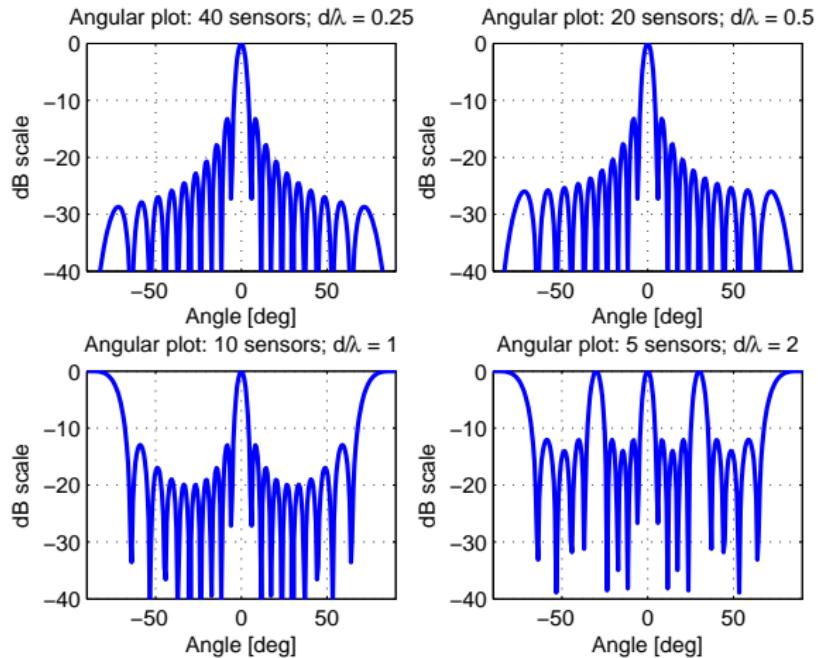
Angular plot: 16 sensors; Aperture  $L = 8\lambda$



*Note for assignment:* Usually maximum at 0 [dB]

# ULA Beampattern: dB scale, variable $d$ , fixed $L$

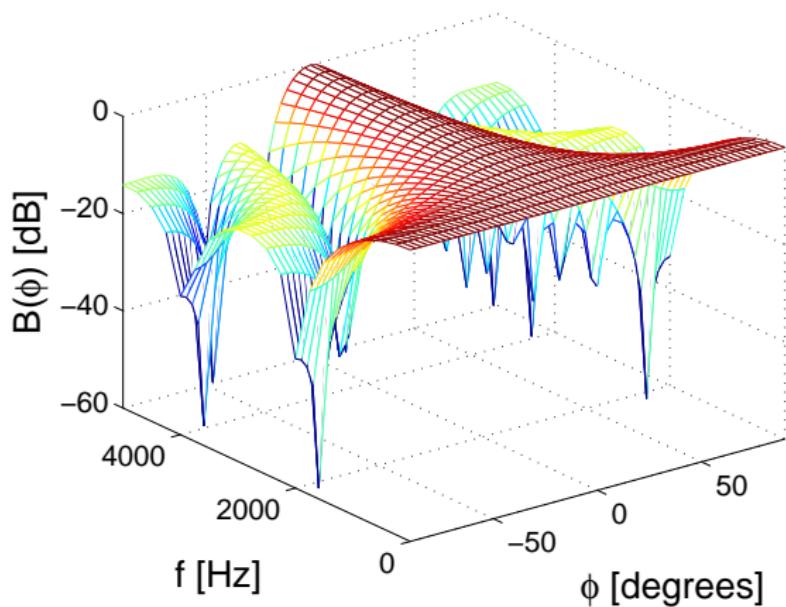
$$\text{Fixed } L = J \cdot d = 10\lambda$$



# ULA Beampattern: Frequency dependency

Example:  $d = \frac{\lambda_{min}}{2} = \frac{c}{2 \cdot f_{max}} \approx 3.5[\text{cm}]$

ULA Far field:  $J=5$ ,  $d=0.035$ ,  $f=0$  to 5000 Hz



## ULA Beampattern: DFT view

With  $\textcolor{blue}{u} = \frac{d \sin(\theta)}{\lambda}$   $\Rightarrow$  ULA beampattern becomes:

$$B(\textcolor{blue}{u}) = \frac{1}{J^2} |\underline{w}^h \cdot \underline{a}(\textcolor{blue}{u})|^2 = \frac{1}{J^2} \left| \sum_{p=0}^{J-1} w_p^* e^{-j2\pi p \textcolor{blue}{u}} \right|^2$$

## ULA Beampattern: DFT view

With  $\textcolor{blue}{u} = \frac{d \sin(\theta)}{\lambda} \Rightarrow$  ULA beampattern becomes:

$$B(\textcolor{blue}{u}) = \frac{1}{J^2} |\underline{w}^h \cdot \underline{a}(\textcolor{blue}{u})|^2 = \frac{1}{J^2} \left| \sum_{p=0}^{J-1} w_p^* e^{-j2\pi p \textcolor{blue}{u}} \right|^2$$

Note: Since unambiguous angles  $-\frac{\pi}{2} \leq \theta \leq \frac{\pi}{2}$ , to avoid aliasing

$$-\frac{1}{2} \leq \textcolor{blue}{u} \leq \frac{1}{2} \Leftrightarrow d \leq \frac{\lambda}{2}$$

## ULA Beampattern: DFT view

With  $\textcolor{blue}{u} = \frac{d \sin(\theta)}{\lambda}$   $\Rightarrow$  ULA beampattern becomes:

$$B(\textcolor{blue}{u}) = \frac{1}{J^2} |\underline{w}^h \cdot \underline{a}(\textcolor{blue}{u})|^2 = \frac{1}{J^2} \left| \sum_{p=0}^{J-1} w_p^* e^{-j 2\pi p \textcolor{blue}{u}} \right|^2$$

Note: Since unambiguous angles  $-\frac{\pi}{2} \leq \theta \leq \frac{\pi}{2}$ , to avoid aliasing

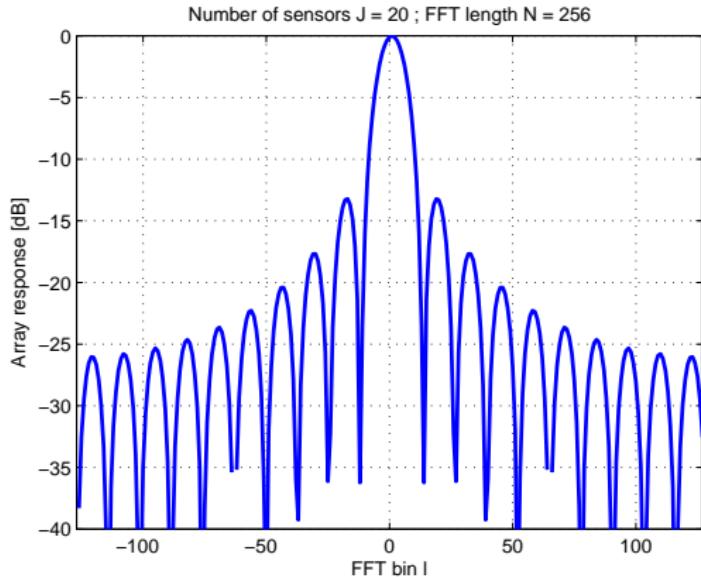
$$-\frac{1}{2} \leq \textcolor{blue}{u} \leq \frac{1}{2} \Leftrightarrow d \leq \frac{\lambda}{2}$$

Zero padded DFT: With  $N \geq J$

$$B_{\textcolor{red}{I}} = \frac{1}{J^2} \left| \sum_{p=0}^{J-1} w_p^* e^{-j \frac{2\pi}{N} p \textcolor{red}{I}} \right|^2$$

with  $\textcolor{red}{I} = N \cdot \textcolor{blue}{u} = N \cdot \frac{d \sin(\theta)}{\lambda}$

# ULA Beampattern: DFT view



Compute corresponding angle via:  $\theta = \arcsin \left( \frac{l}{N} \cdot \frac{\lambda}{d} \right)$

# Data dependent beamforming

# Data dependent beamforming

## Conventional approaches:

- Beam steering
- Tapering
- Null steering
- Array response design

# Beam steering

## **Purpose:**

---

*Compensate propagation path length differences of direct path from source to each sensor resulting in properly aligned direct path signals at the output*

# Beam steering

## **Purpose:**

---

*Compensate propagation path length differences of direct path from source to each sensor resulting in properly aligned direct path signals at the output*

## **Design procedure:**

---

*Design weights such that delays in between sensor elements are compensated  $\Rightarrow$  Beampattern rotates*

# Beam steering

## Purpose:

*Compensate propagation path length differences of direct path from source to each sensor resulting in properly aligned direct path signals at the output*

## Design procedure:

*Design weights such that delays in between sensor elements are compensated  $\Rightarrow$  Beampattern rotates*

*Other name: **Delay and Sum Beamformer (DSB)***

## Beam steering

Desired source signal  $s(t)$ , of single frequency  $f_d = \frac{c}{\lambda}$ , at DOA  $\theta_0$ :

$$\Rightarrow x_i[k] = s[k] \cdot a_i(\theta_0) = s[k] \cdot e^{-j2\pi f_d \tau_i}$$

# Beam steering

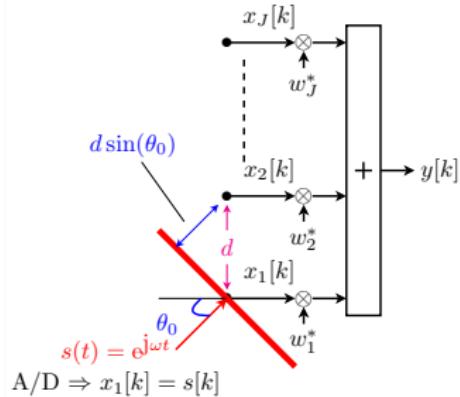
Desired source signal  $s(t)$ , of single frequency  $f_d = \frac{c}{\lambda}$ , at DOA  $\theta_0$ :

$$\Rightarrow x_i[k] = s[k] \cdot a_i(\theta_0) = s[k] \cdot e^{-j2\pi f_d \tau_i}$$

ULA delay at sensor  $i$ :

$$\tau_i = (i - 1) \cdot \frac{d \sin(\theta_0)}{c} \Rightarrow$$

$$x_i[k] = s[k] \cdot e^{-j2\pi(i-1)\frac{d \sin(\theta_0)}{\lambda}}$$



# Beam steering

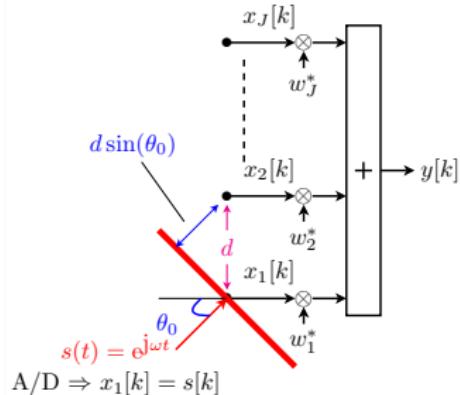
Desired source signal  $s(t)$ , of single frequency  $f_d = \frac{c}{\lambda}$ , at DOA  $\theta_0$ :

$$\Rightarrow x_i[k] = s[k] \cdot a_i(\theta_0) = s[k] \cdot e^{-j2\pi f_d \tau_i}$$

ULA delay at sensor  $i$ :

$$\tau_i = (i - 1) \cdot \frac{d \sin(\theta_0)}{c} \Rightarrow$$

$$x_i[k] = s[k] \cdot e^{-j2\pi(i-1)\frac{d \sin(\theta_0)}{\lambda}}$$



In order to properly align desired source (DOA =  $\theta_0$ ) at output:

$$w_i^* = e^{+j2\pi(i-1)\frac{d \sin(\theta_0)}{\lambda}} \Rightarrow y[k] = J \cdot s[k]$$

## Beam steering

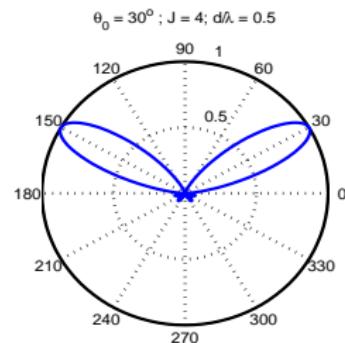
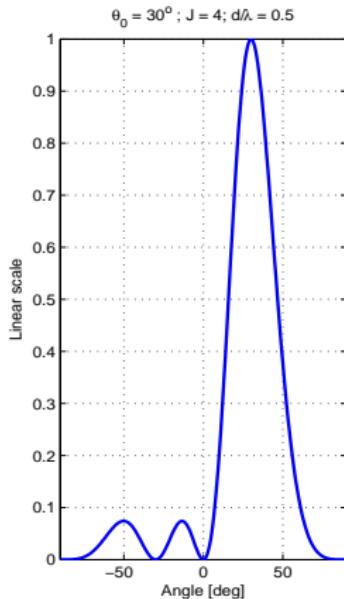
Resulting beampattern shifted/ rotated over  $\theta_0$ :

$$B(\theta) = \frac{1}{J^2} |\underline{w}^h \cdot \underline{a}(\theta)|^2 = \frac{1}{J^2} \left| \sum_{i=1}^J e^{-j2\pi f_d(i-1)\frac{d}{c}(\sin(\theta) - \sin(\theta_0))} \right|^2$$

# Beam steering

Resulting beampattern shifted/ rotated over  $\theta_0$ :

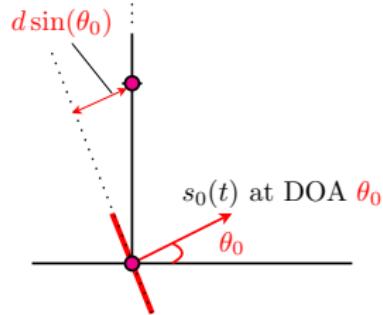
$$B(\theta) = \frac{1}{J^2} |\underline{w}^h \cdot \underline{a}(\theta)|^2 = \frac{1}{J^2} \left| \sum_{i=1}^J e^{-j2\pi f_d(i-1)\frac{d}{c}(\sin(\theta) - \sin(\theta_0))} \right|^2$$



# Beam steering

## Electronic vs mechanical beamsteering

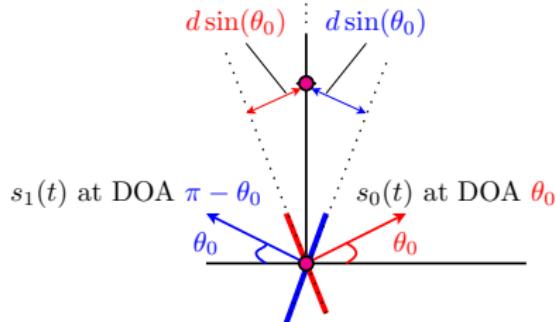
### Electronic beamsteering



# Beam steering

## Electronic vs mechanical beamsteering

### Electronic beamsteering

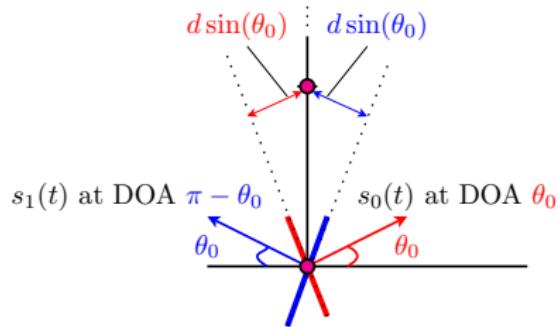


Ambiguity between DOA's  $\theta_0$  and  $\pi - \theta_0$

# Beam steering

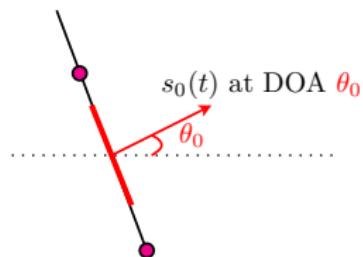
## Electronic vs mechanical beamsteering

### Electronic beamsteering



Ambiguity between DOA's  $\theta_0$  and  $\pi - \theta_0$

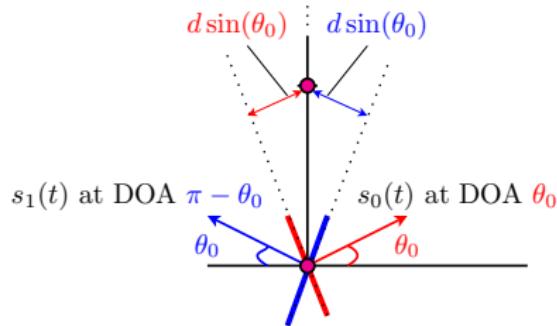
### Mechanical beamsteering



# Beam steering

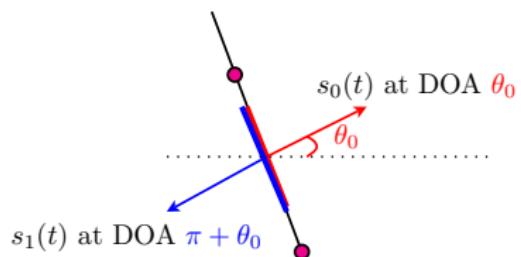
## Electronic vs mechanical beamsteering

### Electronic beamsteering



Ambiguity between DOA's  $\theta_0$  and  $\pi - \theta_0$

### Mechanical beamsteering



Ambiguity between DOA's  $\theta_0$  and  $\pi + \theta_0$

# Beam steering: matched filter

## **Another view to beamsteering for spatially white noise**

## Beam steering: matched filter

### **Another view to beamsteering for spatially white noise**

---

$$\begin{aligned}y[k] &= \underline{\mathbf{w}}^h \cdot \underline{\mathbf{x}}[k] = \underline{\mathbf{w}}^h \cdot (\underline{\mathbf{a}}(\theta)s[k] + \underline{\mathbf{n}}[k]) \\&= \underline{\mathbf{w}}^h \cdot \underline{\mathbf{a}}(\theta)s[k] + \underline{\mathbf{w}}^h \cdot \underline{\mathbf{n}}[k] = y_s[k] + y_n[k]\end{aligned}$$

## Beam steering: matched filter

### **Another view to beamsteering for spatially white noise**

---

$$\begin{aligned}y[k] &= \underline{\mathbf{w}}^h \cdot \underline{\mathbf{x}}[k] = \underline{\mathbf{w}}^h \cdot (\underline{\mathbf{a}}(\theta)s[k] + \underline{\mathbf{n}}[k]) \\&= \underline{\mathbf{w}}^h \cdot \underline{\mathbf{a}}(\theta)s[k] + \underline{\mathbf{w}}^h \cdot \underline{\mathbf{n}}[k] = y_s[k] + y_n[k]\end{aligned}$$

$$\Rightarrow P_y = \sigma_s^2 \cdot \underline{\mathbf{w}}^h \left( \underline{\mathbf{a}}(\theta) \cdot \underline{\mathbf{a}}^h(\theta) \right) \underline{\mathbf{w}} + \sigma_n^2 \cdot \underline{\mathbf{w}}^h \underline{\mathbf{w}} = P_s + P_n$$

# Beam steering: matched filter

## **Another view to beamsteering for spatially white noise**

$$\begin{aligned}y[k] &= \underline{\mathbf{w}}^h \cdot \underline{\mathbf{x}}[k] = \underline{\mathbf{w}}^h \cdot (\underline{\mathbf{a}}(\theta) \mathbf{s}[k] + \underline{\mathbf{n}}[k]) \\&= \underline{\mathbf{w}}^h \cdot \underline{\mathbf{a}}(\theta) \mathbf{s}[k] + \underline{\mathbf{w}}^h \cdot \underline{\mathbf{n}}[k] = y_s[k] + y_n[k]\end{aligned}$$

$$\Rightarrow P_y = \sigma_s^2 \cdot \underline{\mathbf{w}}^h \left( \underline{\mathbf{a}}(\theta) \cdot \underline{\mathbf{a}}^h(\theta) \right) \underline{\mathbf{w}} + \sigma_n^2 \cdot \underline{\mathbf{w}}^h \underline{\mathbf{w}} = P_s + P_n$$

Define in- and output SNR as:  $\text{SNR}_{in} = \frac{\sigma_s^2}{\sigma_n^2}$        $\text{SNR}_o = \frac{P_s}{P_n}$

# Beam steering: matched filter

## **Another view to beamsteering for spatially white noise**

$$\begin{aligned}y[k] &= \underline{\mathbf{w}}^h \cdot \underline{\mathbf{x}}[k] = \underline{\mathbf{w}}^h \cdot (\underline{\mathbf{a}}(\theta) \mathbf{s}[k] + \underline{\mathbf{n}}[k]) \\&= \underline{\mathbf{w}}^h \cdot \underline{\mathbf{a}}(\theta) \mathbf{s}[k] + \underline{\mathbf{w}}^h \cdot \underline{\mathbf{n}}[k] = y_s[k] + y_n[k]\end{aligned}$$

$$\Rightarrow P_y = \sigma_s^2 \cdot \underline{\mathbf{w}}^h \left( \underline{\mathbf{a}}(\theta) \cdot \underline{\mathbf{a}}^h(\theta) \right) \underline{\mathbf{w}} + \sigma_n^2 \cdot \underline{\mathbf{w}}^h \underline{\mathbf{w}} = P_s + P_n$$

Define in- and output SNR as:  $\text{SNR}_{in} = \frac{\sigma_s^2}{\sigma_n^2}$        $\text{SNR}_o = \frac{P_s}{P_n}$

$$\Rightarrow \text{SNR}_o = \frac{\underline{\mathbf{w}}^h \left( \underline{\mathbf{a}}(\theta) \cdot \underline{\mathbf{a}}^h(\theta) \right) \underline{\mathbf{w}}}{\underline{\mathbf{w}}^h \underline{\mathbf{w}}} \cdot \frac{\sigma_s^2}{\sigma_n^2} = G(\theta) \cdot \text{SNR}_{in}$$

# Beam steering: matched filter

## Another view to beamsteering for spatially white noise

$$\begin{aligned}y[k] &= \underline{\mathbf{w}}^h \cdot \underline{\mathbf{x}}[k] = \underline{\mathbf{w}}^h \cdot (\underline{\mathbf{a}}(\theta) \mathbf{s}[k] + \underline{\mathbf{n}}[k]) \\&= \underline{\mathbf{w}}^h \cdot \underline{\mathbf{a}}(\theta) \mathbf{s}[k] + \underline{\mathbf{w}}^h \cdot \underline{\mathbf{n}}[k] = y_s[k] + y_n[k]\end{aligned}$$

$$\Rightarrow P_y = \sigma_s^2 \cdot \underline{\mathbf{w}}^h \left( \underline{\mathbf{a}}(\theta) \cdot \underline{\mathbf{a}}^h(\theta) \right) \underline{\mathbf{w}} + \sigma_n^2 \cdot \underline{\mathbf{w}}^h \underline{\mathbf{w}} = P_s + P_n$$

Define in- and output SNR as:  $\text{SNR}_{in} = \frac{\sigma_s^2}{\sigma_n^2}$        $\text{SNR}_o = \frac{P_s}{P_n}$

$$\Rightarrow \text{SNR}_o = \frac{\underline{\mathbf{w}}^h \left( \underline{\mathbf{a}}(\theta) \cdot \underline{\mathbf{a}}^h(\theta) \right) \underline{\mathbf{w}}}{\underline{\mathbf{w}}^h \underline{\mathbf{w}}} \cdot \frac{\sigma_s^2}{\sigma_n^2} = G(\theta) \cdot \text{SNR}_{in}$$

Thus  $\text{SNR}_o$  maximized by choosing  $\underline{\mathbf{w}} = \beta \cdot \underline{\mathbf{a}}(\theta)$ , with  $\beta$  some constant

# Beam steering: matched filter

## Another view to beamsteering for spatially white noise

$$\begin{aligned}y[k] &= \underline{\mathbf{w}}^h \cdot \underline{\mathbf{x}}[k] = \underline{\mathbf{w}}^h \cdot (\underline{\mathbf{a}}(\theta) \mathbf{s}[k] + \underline{\mathbf{n}}[k]) \\&= \underline{\mathbf{w}}^h \cdot \underline{\mathbf{a}}(\theta) \mathbf{s}[k] + \underline{\mathbf{w}}^h \cdot \underline{\mathbf{n}}[k] = y_s[k] + y_n[k]\end{aligned}$$

$$\Rightarrow P_y = \sigma_s^2 \cdot \underline{\mathbf{w}}^h \left( \underline{\mathbf{a}}(\theta) \cdot \underline{\mathbf{a}}^h(\theta) \right) \underline{\mathbf{w}} + \sigma_n^2 \cdot \underline{\mathbf{w}}^h \underline{\mathbf{w}} = P_s + P_n$$

Define in- and output SNR as:  $\text{SNR}_{in} = \frac{\sigma_s^2}{\sigma_n^2}$        $\text{SNR}_o = \frac{P_s}{P_n}$

$$\Rightarrow \text{SNR}_o = \frac{\underline{\mathbf{w}}^h \left( \underline{\mathbf{a}}(\theta) \cdot \underline{\mathbf{a}}^h(\theta) \right) \underline{\mathbf{w}}}{\underline{\mathbf{w}}^h \underline{\mathbf{w}}} \cdot \frac{\sigma_s^2}{\sigma_n^2} = G(\theta) \cdot \text{SNR}_{in}$$

Thus  $\text{SNR}_o$  maximized by choosing  $\underline{\mathbf{w}} = \beta \cdot \underline{\mathbf{a}}(\theta)$ , with  $\beta$  some constant

*Conclusion:*

Spatial filter that maximizes DOA  $\equiv$  Matched filter ( $= \max \text{SNR}_o$ )

## Beam steering

*Notes on beamsteering (delay and sum beamforming (DSB)):*

---

- ▶ Source location (or DOA) required
- ▶ Position sensors must be known
- ▶ DSB aligns only direct path
- ▶ Difference between electronic vs mechanical beamsteering
- ▶ In sense of max  $\text{SNR}_o$ , spatial matched filter optimum for spatially white noise

## Beam steering: Discrete-time domain

Steering delay  $\tau = \frac{d \sin(\theta)}{c}$ ; Sample rate  $f_s = \frac{1}{T_s} \Rightarrow$

## Beam steering: Discrete-time domain

Steering delay  $\tau = \frac{d \sin(\theta)}{c}$ ; Sample rate  $f_s = \frac{1}{T_s} \Rightarrow$

Steering possible for values  $\tau = \alpha \cdot T_s$  with  $|\alpha| = 0, 1, \dots$

## Beam steering: Discrete-time domain

Steering delay  $\tau = \frac{d \sin(\theta)}{c}$ ; Sample rate  $f_s = \frac{1}{T_s} \Rightarrow$

Steering possible for values  $\tau = \alpha \cdot T_s$  with  $|\alpha| = 0, 1, \dots$

Possible steering to:  $\theta_s = \arcsin\left(\frac{c \cdot \alpha \cdot T_s}{d}\right)$  with  $|\alpha| = 0, 1, \dots$

## Beam steering: Discrete-time domain

Steering delay  $\tau = \frac{d \sin(\theta)}{c}$ ; Sample rate  $f_s = \frac{1}{T_s} \Rightarrow$

Steering possible for values  $\tau = \alpha \cdot T_s$  with  $|\alpha| = 0, 1, \dots$

Possible steering to:  $\theta_s = \arcsin\left(\frac{c \cdot \alpha \cdot T_s}{d}\right)$  with  $|\alpha| = 0, 1, \dots$

### Example:

$$d = \frac{\lambda}{2}, \lambda = \frac{c}{f_0} \text{ and } f_s = 2\gamma f_0 \Rightarrow \theta_s = \arcsin\left(\frac{\alpha}{\gamma}\right) \Leftrightarrow |\alpha| \leq \gamma$$

## Beam steering: Discrete-time domain

Steering delay  $\tau = \frac{d \sin(\theta)}{c}$ ; Sample rate  $f_s = \frac{1}{T_s} \Rightarrow$

Steering possible for values  $\tau = \alpha \cdot T_s$  with  $|\alpha| = 0, 1, \dots$

Possible steering to:  $\theta_s = \arcsin\left(\frac{c \cdot \alpha \cdot T_s}{d}\right)$  with  $|\alpha| = 0, 1, \dots$

### Example:

$$d = \frac{\lambda}{2}, \lambda = \frac{c}{f_0} \text{ and } f_s = 2\gamma f_0 \Rightarrow \theta_s = \arcsin\left(\frac{\alpha}{\gamma}\right) \Leftrightarrow |\alpha| \leq \gamma$$

**Conclusion:** Beam can only be steered to  $1 + 2\lfloor\gamma\rfloor$  different angles!

## Beam steering: Discrete-time domain

Steering delay  $\tau = \frac{d \sin(\theta)}{c}$ ; Sample rate  $f_s = \frac{1}{T_s} \Rightarrow$

Steering possible for values  $\tau = \alpha \cdot T_s$  with  $|\alpha| = 0, 1, \dots$

Possible steering to:  $\theta_s = \arcsin\left(\frac{c \cdot \alpha \cdot T_s}{d}\right)$  with  $|\alpha| = 0, 1, \dots$

### Example:

$$d = \frac{\lambda}{2}, \lambda = \frac{c}{f_0} \text{ and } f_s = 2\gamma f_0 \Rightarrow \theta_s = \arcsin\left(\frac{\alpha}{\gamma}\right) \Leftrightarrow |\alpha| \leq \gamma$$

**Conclusion:** Beam can only be steered to  $1 + 2\lfloor\gamma\rfloor$  different angles!

### Example:

$$f_s = 4 \cdot f_0 \Rightarrow \text{Beam can be steered to } 0^\circ, \pm 30^\circ, \pm 90^\circ$$

## Beam steering: Discrete-time domain

Steering delay  $\tau = \frac{d \sin(\theta)}{c}$ ; Sample rate  $f_s = \frac{1}{T_s} \Rightarrow$

Steering possible for values  $\tau = \alpha \cdot T_s$  with  $|\alpha| = 0, 1, \dots$

Possible steering to:  $\theta_s = \arcsin\left(\frac{c \cdot \alpha \cdot T_s}{d}\right)$  with  $|\alpha| = 0, 1, \dots$

### Example:

$$d = \frac{\lambda}{2}, \lambda = \frac{c}{f_0} \text{ and } f_s = 2\gamma f_0 \Rightarrow \theta_s = \arcsin\left(\frac{\alpha}{\gamma}\right) \Leftrightarrow |\alpha| \leq \gamma$$

**Conclusion:** Beam can only be steered to  $1 + 2[\gamma]$  different angles!

### Example:

$$f_s = 4 \cdot f_0 \Rightarrow \text{Beam can be steered to } 0^\circ, \pm 30^\circ, \pm 90^\circ$$

If more directions needed: Use interpolation and/or fractional delays

# Beam steering: Tapering

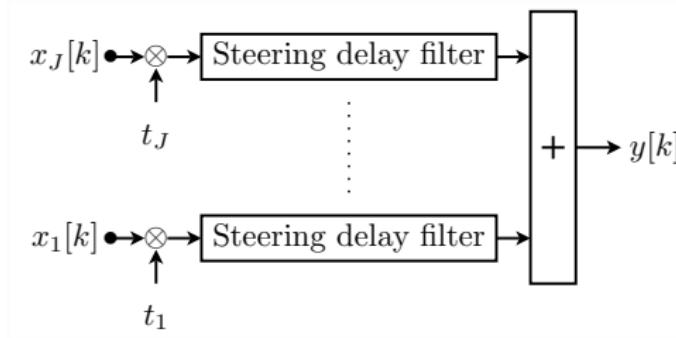
## Goal:

*Control shape of response i.e. to form beam. Thus window (weighted) sensor signals to compromise between resolution (main lobe width) and leakage (side lobe level)  $\Rightarrow \underline{w}_t = \underline{t} \odot \underline{w}$  with  $\underline{t}$  taper window and  $\odot$  element by element multiplication*

# Beam steering: Tapering

## Goal:

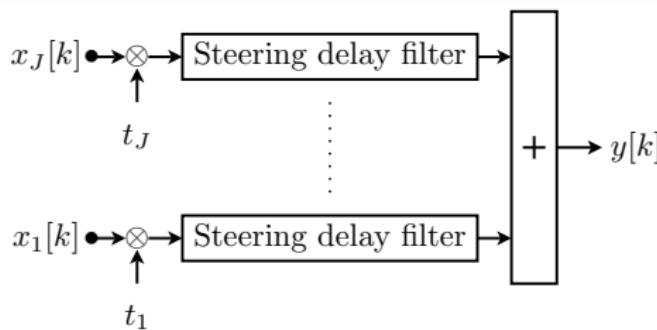
*Control shape of response i.e. to form beam. Thus window (weighted) sensor signals to compromise between resolution (main lobe width) and leakage (side lobe level)  $\Rightarrow \underline{w}_t = \underline{t} \odot \underline{w}$  with  $\underline{t}$  taper window and  $\odot$  element by element multiplication*



# Beam steering: Tapering

## Goal:

*Control shape of response i.e. to form beam. Thus window (weighted) sensor signals to compromise between resolution (main lobe width) and leakage (side lobe level)  $\Rightarrow \underline{w}_t = \underline{t} \odot \underline{w}$  with  $\underline{t}$  taper window and  $\odot$  element by element multiplication*



Notes:

- ▶ Taper weights used to shape beampattern
- ▶ Filters approximate delays (linear phase over frequency band of interest)

## Null-steering

**Goal:** Calculate  $J$  weights to meet  $M$  constraints, e.g. to amplify the desired, and to attenuate the undesired sources

## Null-steering

**Goal:** Calculate  $J$  weights to meet  $M$  constraints, e.g. to amplify the desired, and to attenuate the undesired sources

With  $J \times 1$  weight vector  $\underline{w} = (w_1, \dots, w_J)^t$  set up  $M$  constraints:

$$\underline{a}^h(\omega_1, \theta_1) \cdot \underline{w} = r_d(\omega_1, \theta_1)$$

⋮

$$\Leftrightarrow \boxed{\underline{A}^h \cdot \underline{w} = \underline{r}_d}$$

$$\underline{a}^h(\omega_M, \theta_M) \cdot \underline{w} = r_d(\omega_M, \theta_M)$$

with       $\underline{A} \equiv \underline{A}(\omega, \theta) = (\underline{a}(\omega_1, \theta_1), \dots, \underline{a}(\omega_M, \theta_M))$

$$\underline{r}_d \equiv \underline{r}_d(\omega, \theta) = (r_d(\omega_1, \theta_1), \dots, r_d(\omega_M, \theta_M))^h$$

# Null-steering

**Goal:** Calculate  $J$  weights to meet  $M$  constraints, e.g. to amplify the desired, and to attenuate the undesired sources

With  $J \times 1$  weight vector  $\underline{w} = (w_1, \dots, w_J)^t$  set up  $M$  constraints:

$$\underline{a}^h(\omega_1, \theta_1) \cdot \underline{w} = r_d(\omega_1, \theta_1)$$

⋮

↔

$$\boxed{\underline{A}^h \cdot \underline{w} = \underline{r}_d}$$

$$\underline{a}^h(\omega_M, \theta_M) \cdot \underline{w} = r_d(\omega_M, \theta_M)$$

with       $A \equiv A(\omega, \theta) = (\underline{a}(\omega_1, \theta_1), \dots, \underline{a}(\omega_M, \theta_M))$

$$\underline{r}_d \equiv \underline{r}_d(\omega, \theta) = (r_d(\omega_1, \theta_1), \dots, r_d(\omega_M, \theta_M))^h$$

**Case:**  $M < J$  (*Less constraints than weights*)

$$\boxed{\underline{w} = (A^h)^\dagger \cdot \underline{r}_d = A \cdot (A^h \cdot A)^{-1} \cdot \underline{r}_d}$$

## Null-steering

Example: Null signal at  $90^\circ$  with 2 sensor ULA at distance half wavelength. Thus:  $J = 2$ ,  $M = 1$ ,  $\theta_u = \pi/2$ ,  $r(\theta_u) = 0$ ,  $d/\lambda = 1/2$

## Null-steering

Example: Null signal at  $90^\circ$  with 2 sensor ULA at distance half wavelength. Thus:  $J = 2$ ,  $M = 1$ ,  $\theta_u = \pi/2$ ,  $r(\theta_u) = 0$ ,  $d/\lambda = 1/2$

$$\mathbf{A}^h = \left( 1, e^{j\pi \sin(\pi/2)} \right) = (1, -1) \quad \text{and} \quad \underline{r}_d = (0)$$

## Null-steering

Example: Null signal at  $90^\circ$  with 2 sensor ULA at distance half wavelength. Thus:  $J = 2$ ,  $M = 1$ ,  $\theta_u = \pi/2$ ,  $r(\theta_u) = 0$ ,  $d/\lambda = 1/2$

$$\mathbf{A}^h = \left( 1, e^{j\pi \sin(\pi/2)} \right) = (1, -1) \quad \text{and} \quad \underline{r}_d = (0)$$

$$\Rightarrow \underline{w} = \mathbf{A} \cdot \left( \mathbf{A}^h \cdot \mathbf{A} \right)^{-1} \cdot \underline{r}_d = \dots = \begin{pmatrix} 0 \\ 0 \end{pmatrix}$$

## Null-steering

Example: Null signal at  $90^\circ$  with 2 sensor ULA at distance half wavelength. Thus:  $J = 2$ ,  $M = 1$ ,  $\theta_u = \pi/2$ ,  $r(\theta_u) = 0$ ,  $d/\lambda = 1/2$

$$\mathbf{A}^h = \left( 1, e^{j\pi \sin(\pi/2)} \right) = (1, -1) \quad \text{and} \quad \underline{r}_d = (0)$$

$$\Rightarrow \underline{w} = \mathbf{A} \cdot \left( \mathbf{A}^h \cdot \mathbf{A} \right)^{-1} \cdot \underline{r}_d = \dots = \begin{pmatrix} 0 \\ 0 \end{pmatrix}$$

Indeed nulls at  $90^\circ$  ... however also all others!

## Null-steering

Example: Null signal at  $90^\circ$  with 2 sensor ULA at distance half wavelength. Thus:  $J = 2$ ,  $M = 1$ ,  $\theta_u = \pi/2$ ,  $r(\theta_u) = 0$ ,  $d/\lambda = 1/2$

$$\mathbf{A}^h = \left( 1, e^{j\pi \sin(\pi/2)} \right) = (1, -1) \quad \text{and} \quad \underline{r}_d = (0)$$

$$\Rightarrow \underline{w} = \mathbf{A} \cdot \left( \mathbf{A}^h \cdot \mathbf{A} \right)^{-1} \cdot \underline{r}_d = \dots = \begin{pmatrix} 0 \\ 0 \end{pmatrix}$$

Indeed nulls at  $90^\circ$  ... however also all others!

Note: For this solution we obtain a "line through origin"

$$\mathbf{A}^h \cdot \underline{w} = \underline{r}_d \Leftrightarrow (1, -1) \cdot \begin{pmatrix} w_1 \\ w_2 \end{pmatrix} = 0 \Rightarrow w_1 = w_2$$

Since  $J = 2$  and  $M = 1$ : One degree of freedom left!

This can be used e.g. to overcome the solution  $w_1 = w_2 = 0$

## Null-steering

Example: Two (complex) plane waves. One desired at  $0^\circ$ , other undesired at  $30^\circ$ , ULA with 3 sensors at distance half wavelength  
 $\Rightarrow J = 3, M = 2, \theta_d = 0, \theta_u = \pi/6, r(\theta_d) = 1, r(\theta_u) = 0,$   
 $d/\lambda = 1/2$

## Null-steering

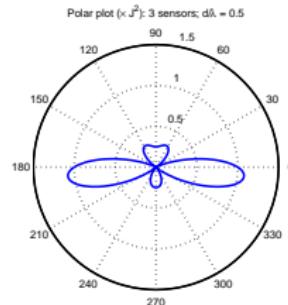
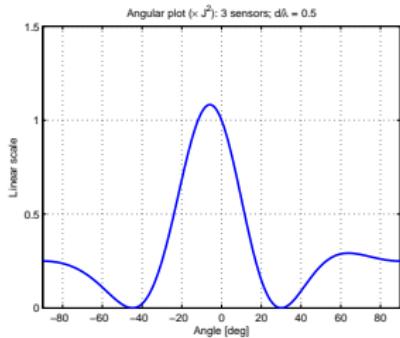
Example: Two (complex) plane waves. One desired at  $0^\circ$ , other undesired at  $30^\circ$ , ULA with 3 sensors at distance half wavelength  
 $\Rightarrow J = 3, M = 2, \theta_d = 0, \theta_u = \pi/6, r(\theta_d) = 1, r(\theta_u) = 0,$   
 $d/\lambda = 1/2 \Rightarrow$

$$A^h = \begin{pmatrix} 1 & 1 & 1 \\ 1 & e^{j\pi \sin(\pi/6)} & e^{j2\pi \sin(\pi/6)} \end{pmatrix}; r_d = (1, 0)^t \Rightarrow \underline{w} = \frac{1}{8} \begin{pmatrix} 3 - j \\ 2 \\ 3 + j \end{pmatrix}$$

# Null-steering

Example: Two (complex) plane waves. One desired at  $0^\circ$ , other undesired at  $30^\circ$ , ULA with 3 sensors at distance half wavelength  
 $\Rightarrow J = 3, M = 2, \theta_d = 0, \theta_u = \pi/6, r(\theta_d) = 1, r(\theta_u) = 0,$   
 $d/\lambda = 1/2 \Rightarrow$

$$A^h = \begin{pmatrix} 1 & 1 & 1 \\ 1 & e^{j\pi \sin(\pi/6)} & e^{j2\pi \sin(\pi/6)} \end{pmatrix}; \underline{r}_d = (1, 0)^t \Rightarrow \underline{w} = \frac{1}{8} \begin{pmatrix} 3 - j \\ 2 \\ 3 + j \end{pmatrix}$$



## Null-steering: Conclusions

*Null steering used to cancel plane waves arriving from **known** directions*

# Null-steering: Conclusions

*Null steering used to cancel plane waves arriving from known directions*

## Required knowledge:

DOA of desired/undesired signals, position of sensors

# Null-steering: Conclusions

*Null steering used to cancel plane waves arriving from **known** directions*

## Required knowledge:

DOA of desired/undesired signals, position of sensors

## Performance:

SNR **not** maximized, but nulls can be put in DOA's of interferences

# Null-steering: Conclusions

*Null steering used to cancel plane waves arriving from known directions*

## Required knowledge:

DOA of desired/undesired signals, position of sensors

## Performance:

SNR **not** maximized, but nulls can be put in DOA's of interferences

## Properties:

- ▶ Result not robust to frequency jammer
- ▶  $J$  weights can set maximum  $J$  predefined conditions
- ▶ Needs much a priori information
- ▶  $M < J$ : Add extra constraints (e.g. minimize output power)
- ▶ Use FIR filters for broadband

## Array response design: $M > J$

Procedure:  $\underline{w} = \arg \min_{\underline{w}} \{E\}$

## Array response design: $M > J$

Procedure:  $\underline{w} = \arg \min_{\underline{w}} \{E\}$

$$\begin{aligned}\text{with error } E &= |\mathbf{A}^h \cdot \underline{w} - \underline{r}_d|^2 = (\underline{w}^h \cdot \mathbf{A} - \underline{r}_d^h) \cdot (\mathbf{A}^h \cdot \underline{w} - \underline{r}_d) \\ &= \underline{w}^h \mathbf{A} \mathbf{A}^h \underline{w} - \underline{w}^h \mathbf{A} \underline{r}_d - \underline{r}_d^h \mathbf{A}^h \underline{w} - \underline{r}_d^h \underline{r}_d\end{aligned}$$

## Array response design: $M > J$

Procedure:  $\underline{w} = \arg \min_{\underline{w}} \{E\}$

$$\begin{aligned}\text{with error } E &= |\mathbf{A}^h \cdot \underline{w} - \underline{r}_d|^2 = (\underline{w}^h \cdot \mathbf{A} - \underline{r}_d^h) \cdot (\mathbf{A}^h \cdot \underline{w} - \underline{r}_d) \\ &= \underline{w}^h \mathbf{A} \mathbf{A}^h \underline{w} - \underline{w}^h \mathbf{A} \underline{r}_d - \underline{r}_d^h \mathbf{A}^h \underline{w} - \underline{r}_d^h \underline{r}_d\end{aligned}$$

$$\frac{dE}{d\underline{w}} = 0 \Rightarrow \boxed{\underline{w} = (\mathbf{A} \cdot \mathbf{A}^h)^{-1} \cdot \mathbf{A} \cdot \underline{r}_d}$$

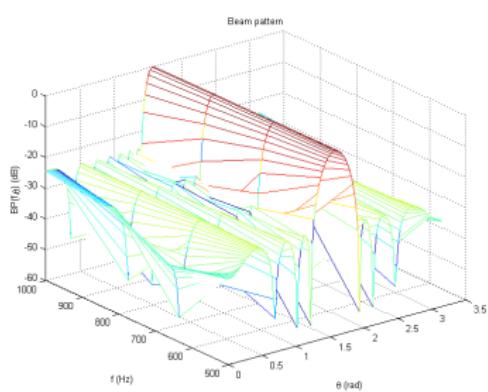
# Array response design: $M > J$

Procedure:  $\underline{w} = \arg \min_{\underline{w}} \{E\}$

$$\begin{aligned}\text{with error } E &= |\mathbf{A}^h \cdot \underline{w} - \underline{r}_d|^2 = (\underline{w}^h \cdot \mathbf{A} - \underline{r}_d^h) \cdot (\mathbf{A}^h \cdot \underline{w} - \underline{r}_d) \\ &= \underline{w}^h \mathbf{A} \mathbf{A}^h \underline{w} - \underline{w}^h \mathbf{A} \underline{r}_d - \underline{r}_d^h \mathbf{A}^h \underline{w} - \underline{r}_d^h \underline{r}_d\end{aligned}$$

$$\frac{dE}{d\underline{w}} = 0 \Rightarrow \boxed{\underline{w} = (\mathbf{A} \cdot \mathbf{A}^h)^{-1} \cdot \mathbf{A} \cdot \underline{r}_d}$$

Example:  $B(\theta) = 1$  at  $\theta = \pi/2$  and  $< -25$  [dB] outside this area



# Beamforming: overview

## Data independent (conventional approach): (Part IB)

- ▶ Beamsteering (DSB, phased array)
- ▶ Tapering
- ▶ Null steering/ Array response design

# Beamforming: overview

## **Data independent (conventional approach): (Part IB)**

- ▶ Beamsteering (DSB, phased array)
- ▶ Tapering
- ▶ Null steering/ Array response design

## **Data dependent (statistical optimum): (Part IC)**

- ▶ Minimum Sidelobe Canceller
- ▶ Wiener
- ▶ Maximum SNR
- ▶ Linear Constraint Minimum Variance
- ▶ Generalized Sidelobe Canceller

## Summary part II

**Far field, narrowband source, direction vector  $\underline{v}$ , at position  $\underline{p}_i$ :**

$$s[k]e^{-j\omega\tau_i} \text{ with } \tau_i = \frac{\underline{v}^t \cdot \underline{p}_i}{c}, \omega = 2\pi f, f = \frac{c}{\lambda}$$

For ULA-case at sensor  $i$ :  $s[k] \cdot a_i(\theta)$  with  $a_i(\theta) = e^{-j2\pi(i-1)\frac{d \sin(\theta)}{\lambda}}$

**Array response/ Angular response/ Directivity pattern:**

$$r(\theta) = \underline{w}^h \cdot \underline{a}(\theta)$$

**Beampattern:**  $B(\theta) = \frac{1}{J^2} |r(\theta)|^2$

For ULA with inter element distance  $d$  and  $\underline{w} = \underline{1}$ :

$$B(\theta) = \frac{1}{J^2} |\underline{1}^t \cdot \underline{a}(\theta)|^2 = \frac{1}{J^2} \left| \frac{1 - e^{-j2\pi\frac{d}{\lambda} \sin(\theta)}}{1 - e^{-j2\pi\frac{d}{\lambda} \sin(\theta)}} \right|^2$$

## Summary part II

**Beamsteering (shifted/ rotated over  $\theta_0$ ):**

$$B(\theta) = \frac{1}{J^2} |\underline{w}^h \cdot \underline{a}(\theta)|^2 = \frac{1}{J^2} \left| \sum_{i=1}^J e^{-j2\pi f_d(i-1)\frac{d}{c}(\sin(\theta) - \sin(\theta_0))} \right|^2$$

**Constrained beamforming:**

With  $J \times 1$  weight vector  $\underline{w} = (w_1, \dots, w_J)^t$  set up  $M$  constraints:

$$\underline{a}^h(\omega_1, \theta_1) \cdot \underline{w} = r_d(\omega_1, \theta_1)$$

⋮

↔

$$\boxed{\mathbf{A}^h \cdot \underline{w} = \underline{r}_d}$$

$$\underline{a}^h(\omega_M, \theta_M) \cdot \underline{w} = r_d(\omega_M, \theta_M)$$

**Null steering ( $M < J$ ):**  $\underline{w} = (\mathbf{A}^h)^\dagger \cdot \underline{r}_d = \mathbf{A} \cdot (\mathbf{A}^h \cdot \mathbf{A})^{-1} \cdot \underline{r}_d$

**Array response design ( $M > J$ ):**  $\underline{w} = (\mathbf{A} \cdot \mathbf{A}^h)^{-1} \cdot \mathbf{A} \cdot \underline{r}_d$

# **DOA + Optimum and Adaptive Array Signal Processing**

**(Part IC)**

# DOA estimation

## Goal:

Estimate Direction Of Arrival (DOA) of sources (and interferences) from noisy observations, in order to locate and/or track these sources

# DOA estimation

## Goal:

Estimate Direction Of Arrival (DOA) of sources (and interferences) from noisy observations, in order to locate and/or track these sources

## Main techniques based on:

1. Maximizing power of steered beamformer
2. High-resolution spectral estimation concepts
3. Employing time-difference of arrival (*not in this course*)

# Conventional DOA

## 1. Maximizing power of steered beamformer:

E.g. noisy observation, one source, J sensors:

$$\underline{x}[k] = \underline{a}(\theta_p) \cdot s[k] + \underline{n}[k] \quad \Rightarrow \quad R_x = \sigma_s^2 \underline{a}(\theta_p) \underline{a}^h(\theta_p) + \sigma_n^2 I$$

$$y[k] = \underline{w}^h \cdot \underline{x}[k] \quad \Rightarrow \quad P_y = E\{|y[k]|^2\} = \underline{w}^h \cdot R_x \cdot \underline{w}$$

# Conventional DOA

## 1. Maximizing power of steered beamformer:

E.g. noisy observation, one source, J sensors:

$$\underline{x}[k] = \underline{a}(\theta_p) \cdot s[k] + \underline{n}[k] \quad \Rightarrow \quad R_x = \sigma_s^2 \underline{a}(\theta_p) \underline{a}^h(\theta_p) + \sigma_n^2 I$$

$$y[k] = \underline{w}^h \cdot \underline{x}[k] \quad \Rightarrow \quad P_y = E\{|y[k]|^2\} = \underline{w}^h \cdot R_x \cdot \underline{w}$$

**Spatial spectrum:**  $P(\theta) = \frac{P_y(\theta)}{\|\underline{w}\|^2}$

# Conventional DOA

## 1. Maximizing power of steered beamformer:

E.g. noisy observation, one source, J sensors:

$$\underline{x}[k] = \underline{a}(\theta_p) \cdot s[k] + \underline{n}[k] \quad \Rightarrow \quad R_x = \sigma_s^2 \underline{a}(\theta_p) \underline{a}^h(\theta_p) + \sigma_n^2 I$$

$$y[k] = \underline{w}^h \cdot \underline{x}[k] \quad \Rightarrow \quad P_y = E\{|y[k]|^2\} = \underline{w}^h \cdot R_x \cdot \underline{w}$$

**Spatial spectrum:**  $P(\theta) = \frac{P_y(\theta)}{\|\underline{w}\|^2}$

Beamsteering:  $\text{Max.}\{P_s/P_n\} \Leftrightarrow \underline{w} \equiv \underline{a}(\theta)$

# Conventional DOA

## 1. Maximizing power of steered beamformer:

E.g. noisy observation, one source, J sensors:

$$\underline{x}[k] = \underline{a}(\theta_p) \cdot s[k] + \underline{n}[k] \quad \Rightarrow \quad R_x = \sigma_s^2 \underline{a}(\theta_p) \underline{a}^h(\theta_p) + \sigma_n^2 I$$

$$y[k] = \underline{w}^h \cdot \underline{x}[k] \quad \Rightarrow \quad P_y = E\{|y[k]|^2\} = \underline{w}^h \cdot R_x \cdot \underline{w}$$

**Spatial spectrum:**  $P(\theta) = \frac{P_y(\theta)}{\|\underline{w}\|^2}$

Beamsteering:  $\text{Max.}\{P_s/P_n\} \Leftrightarrow \underline{w} \equiv \underline{a}(\theta)$

$$P(\theta) = \frac{\underline{a}^h(\theta) R_x \underline{a}(\theta)}{J}$$

# Conventional DOA

## 1. Maximizing power of steered beamformer:

E.g. noisy observation, one source, J sensors:

$$\underline{x}[k] = \underline{a}(\theta_p) \cdot s[k] + \underline{n}[k] \quad \Rightarrow \quad R_x = \sigma_s^2 \underline{a}(\theta_p) \underline{a}^h(\theta_p) + \sigma_n^2 I$$

$$y[k] = \underline{w}^h \cdot \underline{x}[k] \quad \Rightarrow \quad P_y = E\{|y[k]|^2\} = \underline{w}^h \cdot R_x \cdot \underline{w}$$

**Spatial spectrum:**  $P(\theta) = \frac{P_y(\theta)}{\|\underline{w}\|^2}$

Beamsteering:  $\text{Max.}\{P_s/P_n\} \Leftrightarrow \underline{w} \equiv \underline{a}(\theta)$

$$P(\theta) = \frac{\underline{a}^h(\theta) R_x \underline{a}(\theta)}{J}$$

Thus peak in  $P(\theta)$  is DOA location  $p$ !

# Conventional DOA

## 1. Maximizing power of steered beamformer:

E.g. noisy observation, one source, J sensors:

$$\underline{x}[k] = \underline{a}(\theta_p) \cdot s[k] + \underline{n}[k] \quad \Rightarrow \quad R_x = \sigma_s^2 \underline{a}(\theta_p) \underline{a}^h(\theta_p) + \sigma_n^2 I$$

$$y[k] = \underline{w}^h \cdot \underline{x}[k] \quad \Rightarrow \quad P_y = E\{|y[k]|^2\} = \underline{w}^h \cdot R_x \cdot \underline{w}$$

**Spatial spectrum:**  $P(\theta) = \frac{P_y(\theta)}{\|\underline{w}\|^2}$

Beamsteering:  $\text{Max.}\{P_s/P_n\} \Leftrightarrow \underline{w} \equiv \underline{a}(\theta)$

$$P(\theta) = \frac{\underline{a}^h(\theta) R_x \underline{a}(\theta)}{J}$$

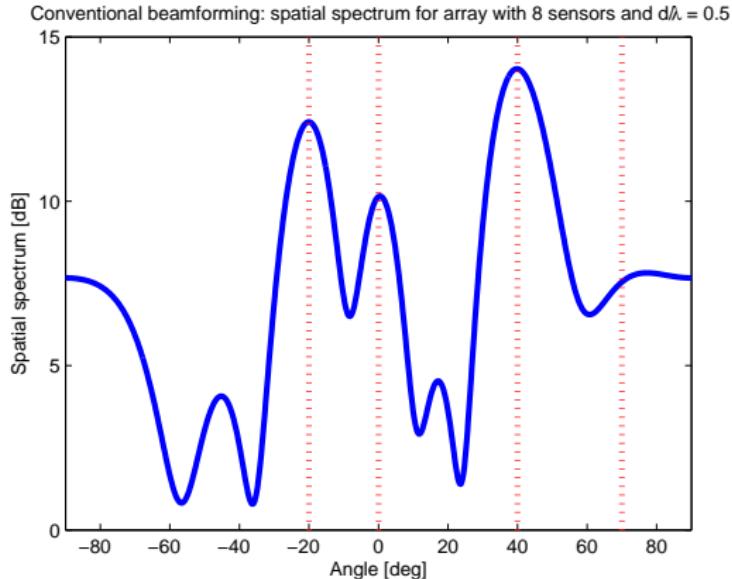
Thus peak in  $P(\theta)$  is DOA location  $p$ !

Note: In practice estimate  $\hat{R}_x = \frac{1}{T} \sum_{k=1}^T \underline{x}[k] \cdot \underline{x}^h[k]$

# Conventional DOA

Example: ULA

$d/\lambda = 0.5$ ;  $P = 4$  sources (at -20, 0, 40 and 70 degrees);  $J = 8$  sensors



## High-resolution DOA

High-resolution DOA technique based on signal subspace (see Appendix) method:

**Spectral MUSIC** = MUltiple Signal Classification

# High-resolution DOA

High-resolution DOA technique based on signal subspace (see Appendix) method:

Spectral MUSIC = MUltiple Signal Classification

Signal model:  $J$  sensors,  $P$  sources,  $P < J$

$$x_i = \sum_{p=1}^P a_i(\theta_p) \cdot s_p[k] + n_i[k] \text{ for } i = 1, \dots, J \quad \Leftrightarrow \quad \underline{x}[k] = \mathbf{A} \cdot \underline{s}[k] + \underline{n}[k]$$

# High-resolution DOA

High-resolution DOA technique based on signal subspace (see Appendix) method:

Spectral MUSIC = MUltiple Signal Classification

Signal model:  $J$  sensors,  $P$  sources,  $P < J$

$$x_i = \sum_{p=1}^P a_i(\theta_p) \cdot s_p[k] + n_i[k] \text{ for } i = 1, \dots, J \quad \Leftrightarrow \quad \underline{x}[k] = \mathbf{A} \cdot \underline{s}[k] + \underline{n}[k]$$

Covariance structure:

$$\mathbf{R}_x = E\{\underline{x} \cdot \underline{x}^h\} = \mathbf{A} \mathbf{R}_s \mathbf{A} + \mathbf{R}_n$$

$$\text{with } \mathbf{R}_s = E\{\underline{s} \cdot \underline{s}^h\} = \text{diag}\{\sigma_{s_1}^2, \dots, \sigma_{s_P}^2\} \quad \text{and} \quad \mathbf{R}_n = \sigma_n^2 \mathbf{I}$$

# High-resolution DOA

*Recall from appendix:*

$$\begin{aligned} R_x &= U_x \Lambda_x U_x^H = U_s \Lambda_{s,n} U_s^H + U_n \Lambda_n U_n^H \\ U_x &= (\underline{u}_1, \dots, \underline{u}_J) ; \Lambda_x = \text{diag}\{\lambda_{x_1}, \dots, \lambda_{x_J}\} \end{aligned}$$

# High-resolution DOA

*Recall from appendix:*

$$\begin{aligned} \mathbf{R}_x &= \mathbf{U}_x \Lambda_x \mathbf{U}_x^h = \mathbf{U}_s \Lambda_{s,n} \mathbf{U}_s^h + \mathbf{U}_n \Lambda_n \mathbf{U}_x^n \\ \mathbf{U}_x &= (\underline{\mathbf{u}}_1, \dots, \underline{\mathbf{u}}_J) ; \Lambda_x = \text{diag}\{\lambda_{x_1}, \dots, \lambda_{x_J}\} \end{aligned}$$

## Signal subspace:

$$\mathbf{U}_s = (\underline{\mathbf{u}}_1, \dots, \underline{\mathbf{u}}_P) ; \Lambda_{s,n} = \text{diag}\{\lambda_{s_1} + \sigma_n^2, \dots, \lambda_{s_P} + \sigma_n^2\}$$

# High-resolution DOA

Recall from appendix:

$$\begin{aligned} \mathbf{R}_x &= \mathbf{U}_x \Lambda_x \mathbf{U}_x^h = \mathbf{U}_s \Lambda_{s,n} \mathbf{U}_s^h + \mathbf{U}_n \Lambda_n \mathbf{U}_n^h \\ \mathbf{U}_x &= (\underline{\mathbf{u}}_1, \dots, \underline{\mathbf{u}}_J) ; \Lambda_x = \text{diag}\{\lambda_{x_1}, \dots, \lambda_{x_J}\} \end{aligned}$$

**Signal subspace:**

$$\mathbf{U}_s = (\underline{\mathbf{u}}_1, \dots, \underline{\mathbf{u}}_P) ; \Lambda_{s,n} = \text{diag}\{\lambda_{s_1} + \sigma_n^2, \dots, \lambda_{s_P} + \sigma_n^2\}$$

**Noise subspace:**

$$\mathbf{U}_n = (\underline{\mathbf{u}}_{P+1}, \dots, \underline{\mathbf{u}}_J) ; \Lambda_n = \text{diag}\{\sigma_n^2, \dots, \sigma_n^2\}$$

# High-resolution DOA

Recall from appendix:

$$\begin{aligned} \mathbf{R}_x &= \mathbf{U}_x \Lambda_x \mathbf{U}_x^h = \mathbf{U}_s \Lambda_{s,n} \mathbf{U}_s^h + \mathbf{U}_n \Lambda_n \mathbf{U}_n^h \\ \mathbf{U}_x &= (\underline{\mathbf{u}}_1, \dots, \underline{\mathbf{u}}_J) ; \Lambda_x = \text{diag}\{\lambda_{x_1}, \dots, \lambda_{x_J}\} \end{aligned}$$

## Signal subspace:

$$\mathbf{U}_s = (\underline{\mathbf{u}}_1, \dots, \underline{\mathbf{u}}_P) ; \Lambda_{s,n} = \text{diag}\{\lambda_{s_1} + \sigma_n^2, \dots, \lambda_{s_P} + \sigma_n^2\}$$

## Noise subspace:

$$\mathbf{U}_n = (\underline{\mathbf{u}}_{P+1}, \dots, \underline{\mathbf{u}}_J) ; \Lambda_n = \text{diag}\{\sigma_n^2, \dots, \sigma_n^2\}$$

## Properties:

$$\mathbf{U}_s \perp \mathbf{U}_n \Leftrightarrow \mathbf{U}_s^h \cdot \mathbf{U}_n = 0 \Leftrightarrow \mathbf{U}_n^h \cdot \mathbf{U}_s = 0$$

# High-resolution DOA

How obtain DOA's from this?

# High-resolution DOA

How obtain DOA's from this?

$$\mathbf{R}_x = \mathbf{U}_s \Lambda_{s,n} \mathbf{U}_s^h + \mathbf{U}_n \Lambda_n \mathbf{U}_n^h$$

$$\mathbf{R}_x = \mathbf{A} \mathbf{R}_s \mathbf{A}^h + \sigma_n^2 \mathbf{I}$$

# High-resolution DOA

How obtain DOA's from this?

$$R_x = U_s \Lambda_{s,n} U_s^h + U_n \Lambda_n U_n^h \Rightarrow R_x U_n = \sigma_n^2 U_n$$

$$R_x = A R_s A^h + \sigma_n^2 I$$

# High-resolution DOA

How obtain DOA's from this?

$$R_x = U_s \Lambda_{s,n} U_s^h + U_n \Lambda_n U_n^h \Rightarrow R_x U_n = \sigma_n^2 U_n$$

$$R_x = A R_s A^h + \sigma_n^2 I \Rightarrow R_x U_n = A R_s A^h U_n + \sigma_n^2 U_n$$

# High-resolution DOA

How obtain DOA's from this?

$$R_x = U_s \Lambda_{s,n} U_s^h + U_n \Lambda_n U_n^h \Rightarrow R_x U_n = \sigma_n^2 U_n$$

$$R_x = A R_s A^h + \sigma_n^2 I \Rightarrow R_x U_n = A R_s A^h U_n + \sigma_n^2 U_n$$

$$\Rightarrow A R_s A^h U_n = 0.$$

# High-resolution DOA

How obtain DOA's from this?

$$R_x = U_s \Lambda_{s,n} U_s^h + U_n \Lambda_n U_n^h \Rightarrow R_x U_n = \sigma_n^2 U_n$$

$$R_x = A R_s A^h + \sigma_n^2 I \Rightarrow R_x U_n = A R_s A^h U_n + \sigma_n^2 U_n$$

$\Rightarrow A R_s A^h U_n = 0$ . Together with  $A R_s$  full rank

$$\Rightarrow A^h U_n = 0 \Leftrightarrow U_n^h A = 0$$

# High-resolution DOA

How obtain DOA's from this?

$$\mathbf{R}_x = \mathbf{U}_s \Lambda_{s,n} \mathbf{U}_s^h + \mathbf{U}_n \Lambda_n \mathbf{U}_n^h \Rightarrow \mathbf{R}_x \mathbf{U}_n = \sigma_n^2 \mathbf{U}_n$$

$$\mathbf{R}_x = \mathbf{A} \mathbf{R}_s \mathbf{A}^h + \sigma_n^2 \mathbf{I} \Rightarrow \mathbf{R}_x \mathbf{U}_n = \mathbf{A} \mathbf{R}_s \mathbf{A}^h \mathbf{U}_n + \sigma_n^2 \mathbf{U}_n$$

$\Rightarrow \mathbf{A} \mathbf{R}_s \mathbf{A}^h \mathbf{U}_n = 0$ . Together with  $\mathbf{A} \mathbf{R}_s$  full rank

$$\Rightarrow \mathbf{A}^h \mathbf{U}_n = 0 \Leftrightarrow \mathbf{U}_n^h \mathbf{A} = 0$$

**Result:** Obtain desired DOA's by solving  $\theta$  from:

$$\underline{\mathbf{u}}_i^h \cdot \underline{\mathbf{a}}(\theta_p) = 0 \quad \text{for} \quad \underline{\mathbf{u}}_i \in \mathbf{U}_n = \{u_{P+1}, \dots, \underline{\mathbf{u}}_J\}$$

$$\underline{\mathbf{a}}(\theta_p) \in \mathbf{A} = \{\underline{\mathbf{a}}(\theta_1), \dots, \underline{\mathbf{a}}(\theta_P)\}$$

# High-resolution DOA

Use this result in "Spectral MUSIC" cost function:

$$C_{SM}(\theta) = \sum_{\underline{u}_i \in U_n} \left| \underline{u}_i^h \underline{a}(\theta) \right|^2 = \underline{a}^h(\theta) \left( \sum_{\underline{u}_i \in U_n} \underline{u}_i \underline{u}_i^h \right) \underline{a}(\theta)$$

# High-resolution DOA

Use this result in "Spectral MUSIC" cost function:

$$\begin{aligned} C_{SM}(\theta) &= \sum_{\underline{u}_i \in U_n} \left| \underline{u}_i^h \underline{a}(\theta) \right|^2 = \underline{a}^h(\theta) \left( \sum_{\underline{u}_i \in U_n} \underline{u}_i \underline{u}_i^h \right) \underline{a}(\theta) \\ &= \underline{a}^h(\theta) \left( U_n U_n^h \right) \underline{a}(\theta) = \underline{a}^h(\theta) (P_n) \underline{a}(\theta) \end{aligned}$$

# High-resolution DOA

Use this result in "Spectral MUSIC" cost function:

$$\begin{aligned} C_{SM}(\theta) &= \sum_{\underline{u}_i \in U_n} \left| \underline{u}_i^h \underline{a}(\theta) \right|^2 = \underline{a}^h(\theta) \left( \sum_{\underline{u}_i \in U_n} \underline{u}_i \underline{u}_i^h \right) \underline{a}(\theta) \\ &= \underline{a}^h(\theta) \left( U_n U_n^h \right) \underline{a}(\theta) = \underline{a}^h(\theta) (P_n) \underline{a}(\theta) \end{aligned}$$

with projection matrix  $P_n = U_n U_n^h$

## High-resolution DOA

Use this result in "Spectral MUSIC" cost function:

$$\begin{aligned} C_{SM}(\theta) &= \sum_{\underline{u}_i \in U_n} \left| \underline{u}_i^h \underline{a}(\theta) \right|^2 = \underline{a}^h(\theta) \left( \sum_{\underline{u}_i \in U_n} \underline{u}_i \underline{u}_i^h \right) \underline{a}(\theta) \\ &= \underline{a}^h(\theta) \left( U_n U_n^h \right) \underline{a}(\theta) = \underline{a}^h(\theta) (P_n) \underline{a}(\theta) \end{aligned}$$

with projection matrix  $P_n = U_n U_n^h$

$P_n \underline{a}(\theta)$  is projection of  $\underline{a}(\theta)$  on noise subspace  $U_n$ .

## High-resolution DOA

Use this result in "Spectral MUSIC" cost function:

$$\begin{aligned} C_{SM}(\theta) &= \sum_{\underline{u}_i \in U_n} \left| \underline{u}_i^h \underline{a}(\theta) \right|^2 = \underline{a}^h(\theta) \left( \sum_{\underline{u}_i \in U_n} \underline{u}_i \underline{u}_i^h \right) \underline{a}(\theta) \\ &= \underline{a}^h(\theta) \left( U_n U_n^h \right) \underline{a}(\theta) = \underline{a}^h(\theta) (P_n) \underline{a}(\theta) \end{aligned}$$

with projection matrix  $P_n = U_n U_n^h$

$P_n \underline{a}(\theta)$  is projection of  $\underline{a}(\theta)$  on noise subspace  $U_n$ .

**Conclusion:**  $C_{SM}$  is innerproduct of  $\underline{a}(\theta)$  and projection of  $\underline{a}(\theta)$  on  $U_n$

$C_{SM} = 0$  only true for DOA's  $\theta_p$  with  $p = 1, \dots, P$

# High-resolution DOA

Define pseudo-spectrum:

$$P_{SM}(\theta) = \frac{||\underline{a}(\theta)||^2}{C_{SM}} = \frac{||\underline{a}(\theta)||^2}{\underline{a}^h(\theta) \mathbf{U}_n \mathbf{U}_n^h \underline{a}(\theta)} = \frac{J}{\underline{a}^h(\theta) \mathbf{P}_n \underline{a}(\theta)}$$

# High-resolution DOA

Define pseudo-spectrum:

$$P_{SM}(\theta) = \frac{||\underline{a}(\theta)||^2}{C_{SM}} = \frac{||\underline{a}(\theta)||^2}{\underline{a}^h(\theta) \mathbf{U}_n \mathbf{U}_n^h \underline{a}(\theta)} = \frac{J}{\underline{a}^h(\theta) \mathbf{P}_n \underline{a}(\theta)}$$

Notes:

- ▶ Minimizing  $C_{SM}(\theta)$   $\Leftrightarrow$  Maximizing  $P_{SM}(\theta)$
- ▶ Sharp peaks (high resolution) in vicinity of source DOA's  $\theta_p$ ,  $p = 1, \dots, P$
- ▶ In practice:  $\hat{\mathbf{R}}_x \Rightarrow \hat{\mathbf{U}}_s$  not completely orthogonal to  $\hat{\mathbf{U}}_n$
- ▶  $P_{SM}$  averages  $J - P$  pseudo spectra of individual noise vectors
- ▶ 'Pseudo' in name since no info about real power

# High-resolution DOA

## Spectral-MUSIC algorithm

# High-resolution DOA

## Spectral-MUSIC algorithm

1. Compute/ estimate  $R_x$

# High-resolution DOA

## Spectral-MUSIC algorithm

1. Compute/ estimate  $R_x$
2. Compute EVD of  $R_x$  and split signal- noise subspace:

$$R_x = U_x \Lambda_x U_x^h = U_s \Lambda_{s,n} U_s^h + U_n \Lambda_n U_n^h$$

# High-resolution DOA

## Spectral-MUSIC algorithm

1. Compute/ estimate  $R_x$
2. Compute EVD of  $R_x$  and split signal- noise subspace:

$$R_x = U_x \Lambda_x U_x^h = U_s \Lambda_{s,n} U_s^h + U_n \Lambda_n U_n^h$$

3. Compute projection matrix:  $P_n = U_n U_n^h$

# High-resolution DOA

## Spectral-MUSIC algorithm

1. Compute/ estimate  $R_x$
2. Compute EVD of  $R_x$  and split signal- noise subspace:

$$R_x = U_x \Lambda_x U_x^h = U_s \Lambda_{s,n} U_s^h + U_n \Lambda_n U_n^h$$

3. Compute projection matrix:  $P_n = U_n U_n^h$
4. Evaluate pseudo spectrum:

$$P_{SM}(\theta) = \frac{J}{\underline{a}^h(\theta) P_n \underline{a}(\theta)}$$

# High-resolution DOA

## Spectral-MUSIC algorithm

1. Compute/ estimate  $R_x$
2. Compute EVD of  $R_x$  and split signal- noise subspace:

$$R_x = U_x \Lambda_x U_x^h = U_s \Lambda_{s,n} U_s^h + U_n \Lambda_n U_n^h$$

3. Compute projection matrix:  $P_n = U_n U_n^h$
4. Evaluate pseudo spectrum:

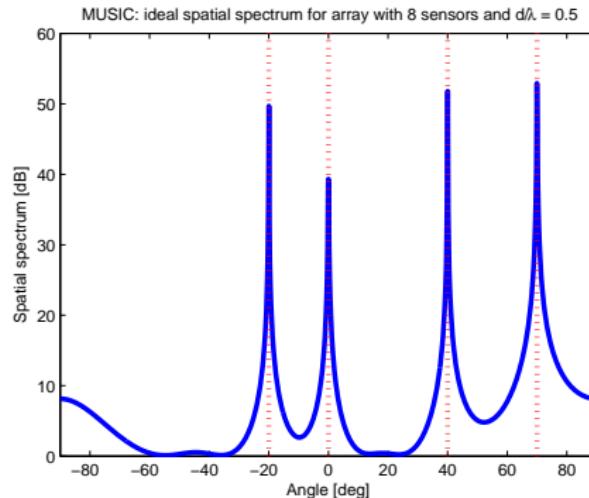
$$P_{SM}(\theta) = \frac{J}{\underline{a}^h(\theta) P_n \underline{a}(\theta)}$$

5. Source DOA's  $\theta_p$  for  $p = 1, \dots, P$ :  
Locate  $P$  sharpest peaks in  $P_{SM}(\theta)$

# High-resolution DOA

Example: ULA

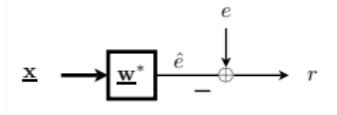
$d/\lambda = 0.5$ ;  $P = 4$  sources (at -20, 0, 40 and 70 degrees);  $J = 8$  sensors



*Note: This example exploits  $J - P = 4$  noise sources*

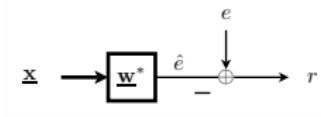
# Minimum Mean Squared Error (Wiener)

# Minimum Mean Squared Error (Wiener)



$$\begin{aligned}\underline{w}_{mse} &= \operatorname{argmin}_{\underline{w}} \{J\} \\ J &= E\{|r|^2\}\end{aligned}$$

# Minimum Mean Squared Error (Wiener)

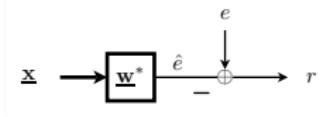


$$\begin{aligned}\underline{w}_{mse} &= \operatorname{argmin}_{\underline{w}} \{J\} \\ J &= E\{|r|^2\}\end{aligned}$$

$$\Rightarrow \underline{w}_{mse} = \underline{R}_x^{-1} \underline{r}_{e^*x} \quad \text{and} \quad J_{min} = E\{|e|^2\} - \underline{r}_{e^*x}^h \underline{R}_x^{-1} \underline{r}_{e^*x}$$

$$\text{with } \underline{R}_x = E\{\underline{x} \cdot \underline{x}^h\} \quad \underline{r}_{e^*x} = E\{\underline{x} \cdot e^*\}$$

# Minimum Mean Squared Error (Wiener)



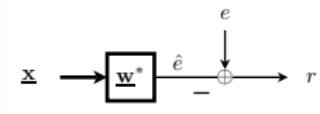
$$\begin{aligned}\underline{w}_{mse} &= \operatorname{argmin}_{\underline{w}} \{J\} \\ J &= E\{|r|^2\}\end{aligned}$$

$$\Rightarrow \underline{w}_{mse} = \underline{R}_x^{-1} \underline{r}_{e^*x} \quad \text{and} \quad J_{min} = E\{|e|^2\} - \underline{r}_{e^*x}^h \underline{R}_x^{-1} \underline{r}_{e^*x}$$

$$\text{with } \underline{R}_x = E\{\underline{x} \cdot \underline{x}^h\} \quad \underline{r}_{e^*x} = E\{\underline{x} \cdot e^*\}$$

**Necessary knowledge:**  $\underline{R}_x$  and  $\underline{r}_{e^*x}$  (both from measurements)

# Minimum Mean Squared Error (Wiener)



$$\underline{w}_{mse} = \operatorname{argmin}_{\underline{w}} \{J\}$$
$$J = E\{|r|^2\}$$

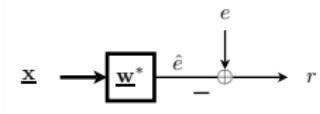
$$\Rightarrow \underline{w}_{mse} = R_x^{-1} \underline{r}_{e^* x} \quad \text{and} \quad J_{min} = E\{|e|^2\} - \underline{r}_{e^* x}^h R_x^{-1} \underline{r}_{e^* x}$$

$$\text{with } R_x = E\{\underline{x} \cdot \underline{x}^h\} \quad \underline{r}_{e^* x} = E\{\underline{x} \cdot e^*\}$$

**Necessary knowledge:**  $R_x$  and  $\underline{r}_{e^* x}$  (both from measurements)

**Example:** ULA, 1 source, narrowband, farfield  $\rightarrow \underline{x} = \underline{a} \cdot s + \underline{n}$  and  $e = s$

# Minimum Mean Squared Error (Wiener)



$$\underline{w}_{mse} = \operatorname{argmin}_{\underline{w}} \{J\}$$
$$J = E\{|r|^2\}$$

$$\Rightarrow \underline{w}_{mse} = R_x^{-1} \underline{r}_{e^* x} \quad \text{and} \quad J_{min} = E\{|e|^2\} - \underline{r}_{e^* x}^h R_x^{-1} \underline{r}_{e^* x}$$

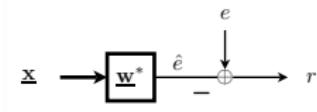
$$\text{with } R_x = E\{\underline{x} \cdot \underline{x}^h\} \quad \underline{r}_{e^* x} = E\{\underline{x} \cdot e^*\}$$

**Necessary knowledge:**  $R_x$  and  $\underline{r}_{e^* x}$  (both from measurements)

**Example:** ULA, 1 source, narrowband, farfield  $\rightarrow \underline{x} = \underline{a} \cdot s + \underline{n}$  and  $e = s$

$$R_x = (\underline{a} \cdot \underline{a}^h) \cdot \sigma_s^2 + \sigma_n^2 \cdot I \quad \text{and} \quad \underline{r}_{e^* x} = \underline{a} \cdot \sigma_s^2$$

# Minimum Mean Squared Error (Wiener)



$$\underline{w}_{mse} = \operatorname{argmin}_{\underline{w}} \{J\}$$
$$J = E\{|r|^2\}$$

$$\Rightarrow \underline{w}_{mse} = R_x^{-1} r_{e^*x} \quad \text{and} \quad J_{min} = E\{|e|^2\} - r_{e^*x}^h R_x^{-1} r_{e^*x}$$

$$\text{with } R_x = E\{\underline{x} \cdot \underline{x}^h\} \quad r_{e^*x} = E\{\underline{x} \cdot e^*\}$$

**Necessary knowledge:**  $R_x$  and  $r_{e^*x}$  (both from measurements)

**Example:** ULA, 1 source, narrowband, farfield  $\rightarrow \underline{x} = \underline{a} \cdot s + \underline{n}$  and  $e = s$

$$R_x = (\underline{a} \cdot \underline{a}^h) \cdot \sigma_s^2 + \sigma_n^2 \cdot I \quad \text{and} \quad r_{e^*x} = \underline{a} \cdot \sigma_s^2$$

$$\Rightarrow \underline{w}_{mse} = \beta \cdot \underline{a} \quad \text{and} \quad J_{min} = \beta \cdot \sigma_n^2 \quad \text{with} \quad \beta = \frac{(\sigma_s^2 / \sigma_n^2)}{1 + J \cdot (\sigma_s^2 / \sigma_n^2)}$$

## Minimum Mean Squared Error (Wiener)

Thus for ULA, one source, narrowband, farfield:

- ▶  $\underline{w}_{mse} = \beta \cdot \underline{a}$ , which is equivalent to matched filter result, which maximizes SNR
- ▶  $J_{min} \approx \frac{1}{J} \cdot \sigma_n^2 \Rightarrow$  SNR improved approx. by factor  $J$

# Minimum Mean Squared Error (Wiener)

Thus for ULA, one source, narrowband, farfield:

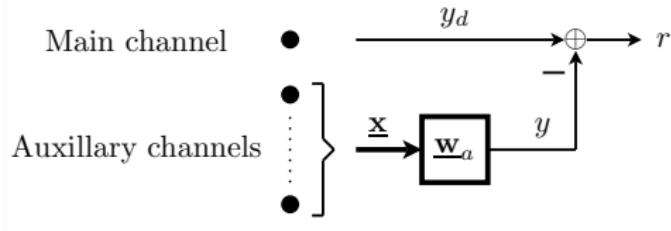
- ▶  $\underline{w}_{mse} = \beta \cdot \underline{a}$ , which is equivalent to matched filter result, which maximizes SNR
- ▶  $J_{min} \approx \frac{1}{J} \cdot \sigma_n^2 \Rightarrow$  SNR improved approx. by factor  $J$

## Conclusion MMSE

- + Simple
- + Direction of desired signal may be unknown
- Must generate reference signal

# Multiple Sidelobe Canceller (Applebaum)

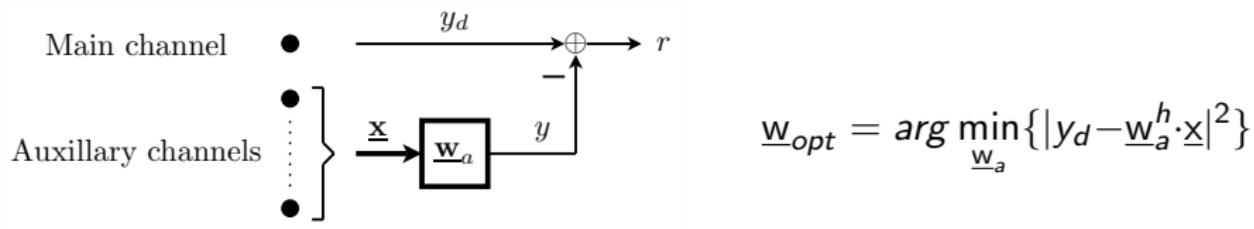
*Use auxillary channel to cancel interference in main channel*



$$\underline{w}_{opt} = \arg \min_{\underline{w}_a} \{ |y_d - \underline{w}_a^h \cdot \underline{x}|^2 \}$$

# Multiple Sidelobe Canceller (Applebaum)

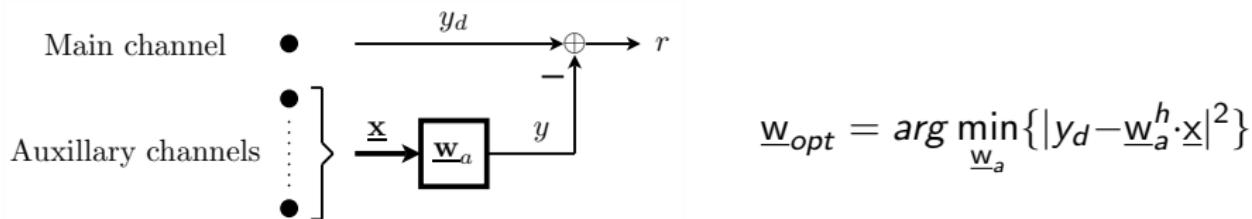
*Use auxillary channel to cancel interference in main channel*



**Main assumption MSC:** Interference assumed to be present in both main and auxillary channels. Desired signal strongly present in main channel, but **below noise level** in auxillary channels

# Multiple Sidelobe Canceller (Applebaum)

*Use auxillary channel to cancel interference in main channel*

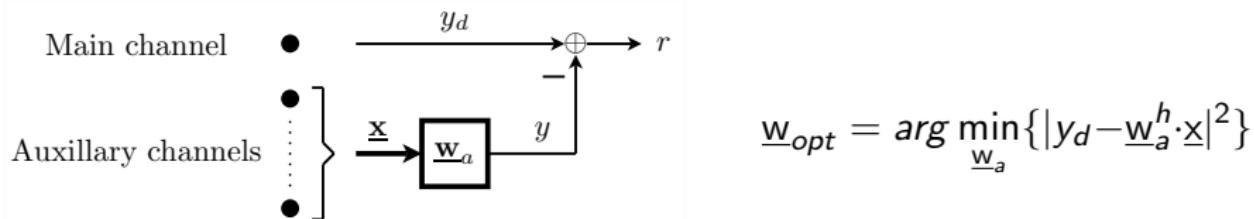


**Main assumption MSC:** Interference assumed to be present in both main and auxillary channels. Desired signal strongly present in main channel, but **below noise level** in auxillary channels

$$\Rightarrow \quad \underline{w}_{opt} = \underline{R}_x^{-1} \cdot \underline{r}_{xy_d}^* \quad \text{and} \quad P_{out} = \sigma_{y_d}^2 - \underline{r}_{xy_d}^h \underline{R}_x^{-1} \underline{r}_{xy_d}$$

# Multiple Sidelobe Canceller (Applebaum)

*Use auxillary channel to cancel interference in main channel*



**Main assumption MSC:** Interference assumed to be present in both main and auxillary channels. Desired signal strongly present in main channel, but **below noise level** in auxillary channels

$$\Rightarrow \quad \underline{w}_{opt} = R_x^{-1} \cdot r_{xy_d}^* \quad \text{and} \quad P_{out} = \sigma_{y_d}^2 - r_{xy_d}^h R_x^{-1} r_{xy_d}$$

**Conclusion MSC:** Simple, but requires desired signal below noise level in auxillary channels, otherwise ...

## Linear Constrained Minimum Variance (Frost)

Previous methods may be unsatisfactory e.g. if desired signal is of unknown strength or is always present →

- ▶ **MSC**: Signal cancelling
- ▶ **Max SINR**: Needs signal and noise covariance estimate
- ▶ **MMSE**: Lack of knowledge reference signal

# Linear Constrained Minimum Variance (Frost)

Previous methods may be unsatisfactory e.g. if desired signal is of unknown strength or is always present →

- ▶ **MSC:** Signal cancelling
- ▶ **Max SINR:** Needs signal and noise covariance estimate
- ▶ **MMSE:** Lack of knowledge reference signal

## Design philosophy LCMV:

*Design weight vector by minimizing average output power subject to M constraints that filter response remains constant at some specific frequencies of interest*

# Linear Constrained Minimum Variance

**Average output power:**

$$P_y = E\{|y|^2\} = \underline{w}^h \cdot E\{\underline{x}\underline{x}^h\} \cdot \underline{w} = \underline{w}^h \cdot R_x \cdot \underline{w}$$

# Linear Constrained Minimum Variance

Average output power:

$$P_y = E\{|y|^2\} = \underline{w}^h \cdot E\{\underline{x}\underline{x}^h\} \cdot \underline{w} = \underline{w}^h \cdot R_x \cdot \underline{w}$$

$M < J$  linear independent constraints:

$C^h \underline{w} = \underline{r}_d$  with  $J \times M$  constraint matrix C

# Linear Constrained Minimum Variance

Average output power:

$$P_y = E\{|y|^2\} = \underline{w}^h \cdot E\{\underline{x}\underline{x}^h\} \cdot \underline{w} = \underline{w}^h \cdot R_x \cdot \underline{w}$$

$M < J$  linear independent constraints:

$C^h \underline{w} = \underline{r}_d$  with  $J \times M$  constraint matrix C

Error criterion:

$$\min_{\underline{w}} \{P_y\} = \min_{\underline{w}} \{\underline{w}^h \cdot R_x \cdot \underline{w}\} \text{ subject to } C^h \cdot \underline{w} = \underline{r}_d$$

# Linear Constrained Minimum Variance

## Average output power:

$$P_y = E\{|y|^2\} = \underline{w}^h \cdot E\{\underline{x}\underline{x}^h\} \cdot \underline{w} = \underline{w}^h \cdot R_x \cdot \underline{w}$$

## $M < J$ linear independent constraints:

$C^h \underline{w} = \underline{r}_d$  with  $J \times M$  constraint matrix C

## Error criterion:

$$\min_{\underline{w}} \{P_y\} = \min_{\underline{w}} \{\underline{w}^h \cdot R_x \cdot \underline{w}\} \text{ subject to } C^h \cdot \underline{w} = \underline{r}_d$$

## Solution via Lagrange multipliers:

$$J = \underline{w}^h \cdot R_x \cdot \underline{w} + \lambda \left( C^h \underline{w} - \underline{r}_d \right)$$

# Linear Constrained Minimum Variance

## Average output power:

$$P_y = E\{|y|^2\} = \underline{w}^h \cdot E\{\underline{x}\underline{x}^h\} \cdot \underline{w} = \underline{w}^h \cdot R_x \cdot \underline{w}$$

## $M < J$ linear independent constraints:

$C^h \underline{w} = \underline{r}_d$  with  $J \times M$  constraint matrix C

## Error criterion:

$$\min_{\underline{w}} \{P_y\} = \min_{\underline{w}} \{\underline{w}^h \cdot R_x \cdot \underline{w}\} \text{ subject to } C^h \cdot \underline{w} = \underline{r}_d$$

## Solution via Lagrange multipliers:

$$J = \underline{w}^h \cdot R_x \cdot \underline{w} + \lambda \left( C^h \underline{w} - \underline{r}_d \right)$$

Solution similar to results in part I:

$$\underline{w} = R_x^{-1} C \left( C^h R_x^{-1} C \right)^{-1} \underline{r}_d$$

and

$$P_y = \underline{r}_d^h \left( C^h R_x^{-1} C \right)^{-1} \underline{r}_d$$

# Linear Constrained Minimum Variance

## Conclusions LCMV:

# Linear Constrained Minimum Variance

## Conclusions LCMV:

- + High resolution

# Linear Constrained Minimum Variance

## Conclusions LCMV:

- + High resolution
- + LCMV controls spectral leakage, in contrast to conventional (e.g. null-steering) methods

# Linear Constrained Minimum Variance

## Conclusions LCMV:

- + High resolution
- + LCMV controls spectral leakage, in contrast to conventional (e.g. null-steering) methods
- + If there are interferences present, LCMV tends to null them out

# Linear Constrained Minimum Variance

## Conclusions LCMV:

- + High resolution
- + LCMV controls spectral leakage, in contrast to conventional (e.g. null-steering) methods
- + If there are interferences present, LCMV tends to null them out
- Sensitive due to inverse correlation matrix

# Minimum Variance Distortionless Response

## MVDR:

$$\min_{\underline{w}} \{P_y\} = \min_{\underline{w}} \{E\{|y|^2\}} \text{ subject to } \underline{w}^h \cdot \underline{a} = 1$$

# Minimum Variance Distortionless Response

MVDR:

$$\min_{\underline{w}} \{P_y\} = \min_{\underline{w}} \{E\{|y|^2\}} \text{ subject to } \underline{w}^h \cdot \underline{a} = 1$$

$$\Rightarrow \boxed{\underline{w} = \frac{\underline{R}^{-1}\underline{a}}{\underline{a}^h \underline{R}^{-1} \underline{a}}} \text{ and } \boxed{P_y = \frac{1}{\underline{a}^h \underline{R}^{-1} \underline{a}}}$$

# Minimum Variance Distortionless Response

## MVDR:

$$\min_{\underline{w}} \{P_y\} = \min_{\underline{w}} \{E\{|y|^2\}} \text{ subject to } \underline{w}^h \cdot \underline{a} = 1$$

$$\Rightarrow \boxed{\underline{w} = \frac{\underline{R}^{-1}\underline{a}}{\underline{a}^h \underline{R}^{-1} \underline{a}}} \text{ and } \boxed{P_y = \frac{1}{\underline{a}^h \underline{R}^{-1} \underline{a}}}$$

## Notes MVDR (=Capon):

- ▶ MVDR is special case of LCMV with  $r_d = 1$  and  $C = \underline{a}$
- ▶ MVDR is max SNR if  $R = \sigma_s^2 \underline{a} \underline{a}^h + R_n$

# Minimum Variance Distortionless Response

## MVDR:

$$\min_{\underline{w}} \{P_y\} = \min_{\underline{w}} \{E\{|y|^2\}} \text{ subject to } \underline{w}^h \cdot \underline{a} = 1$$

$$\Rightarrow \boxed{\underline{w} = \frac{\underline{R}^{-1}\underline{a}}{\underline{a}^h \underline{R}^{-1} \underline{a}}} \text{ and } \boxed{P_y = \frac{1}{\underline{a}^h \underline{R}^{-1} \underline{a}}}$$

## Notes MVDR (=Capon):

- ▶ MVDR is special case of LCMV with  $r_d = 1$  and  $C = \underline{a}$
- ▶ MVDR is max SNR if  $R = \sigma_s^2 \underline{a} \underline{a}^h + R_n$

## Conclusions MVDR/ LCMV:

- + Flexible and general constraints possible
- Computation constraint weight vector not trivial
- As signal extractor: sensitive to errors in DOA
- Problems with correlated signals

## Generalized Sidelobe Canceller (Griffith-Jim)

**GSC:** Alternative formulation of LCMV, illustrates relationship between MSC and LCMV. Mechanism to change constrained minimization problem in unconstrained one

## Generalized Sidelobe Canceller (Griffith-Jim)

**GSC:** Alternative formulation of LCMV, illustrates relationship between MSC and LCMV. Mechanism to change constrained minimization problem in unconstrained one

With  $M$  independent constraints and  $J$  ( $< M$ ) weights  $\Rightarrow$

$$C^h \underline{w} = \underline{r}_d$$

## Generalized Sidelobe Canceller (Griffith-Jim)

GSC: Alternative formulation of LCMV, illustrates relationship between MSC and LCMV. Mechanism to change constrained minimization problem in unconstrained one

With  $M$  independent constraints and  $J (< M)$  weights  $\Rightarrow$

$$C^h \underline{w} = \underline{r}_d$$

Construct full rank  $J \times J$  matrix  $B = (C|T)$

$B$  has  $J$  independent columns; Rank  $J \times M$  matrix  $C$  is  $M$  and  
Rank  $J \times (J - M)$  matrix  $T$  is  $J - M$

## Generalized Sidelobe Canceller (Griffith-Jim)

GSC: Alternative formulation of LCMV, illustrates relationship between MSC and LCMV. Mechanism to change constrained minimization problem in unconstrained one

With  $M$  independent constraints and  $J (< M)$  weights  $\Rightarrow$

$$C^h \underline{w} = \underline{r}_d$$

Construct full rank  $J \times J$  matrix  $B = (C|T)$

$B$  has  $J$  independent columns; Rank  $J \times M$  matrix  $C$  is  $M$  and  
Rank  $J \times (J - M)$  matrix  $T$  is  $J - M$

Any  $\underline{w} \in J$  dimensional space spanned by columns  $B$ :

$\underline{w} = C \cdot \underline{v} - T \cdot \underline{w}_a$  with  $C \cdot \underline{v} = \underline{w}_c$  weights belong to constraint

$$\text{Constraint } \Rightarrow C^h \underline{w} = C^h C \underline{v} - C^h T \underline{w}_a = \underline{r}_d$$

## Generalized Sidelobe Canceller (Griffith-Jim)

GSC: Alternative formulation of LCMV, illustrates relationship between MSC and LCMV. Mechanism to change constrained minimization problem in unconstrained one

With  $M$  independent constraints and  $J (< M)$  weights  $\Rightarrow$

$$C^h \underline{w} = \underline{r}_d$$

Construct full rank  $J \times J$  matrix  $B = (C|T)$

$B$  has  $J$  independent columns; Rank  $J \times M$  matrix  $C$  is  $M$  and  
Rank  $J \times (J - M)$  matrix  $T$  is  $J - M$

Any  $\underline{w} \in J$  dimensional space spanned by columns  $B$ :

$\underline{w} = C \cdot \underline{v} - T \cdot \underline{w}_a$  with  $C \cdot \underline{v} = \underline{w}_c$  weights belong to constraint

$$\text{Constraint } \Rightarrow C^h \underline{w} = C^h C \underline{v} - C^h T \underline{w}_a = \underline{r}_d$$

Construct  $T$  such that  $C^h T = 0 \Rightarrow \underline{v} = (C^h C)^{-1} \underline{r}_d$

## Generalized Sidelobe Canceller (Griffith-Jim)

GSC: Alternative formulation of LCMV, illustrates relationship between MSC and LCMV. Mechanism to change constrained minimization problem in unconstrained one

With  $M$  independent constraints and  $J (< M)$  weights  $\Rightarrow$

$$C^h \underline{w} = \underline{r}_d$$

Construct full rank  $J \times J$  matrix  $B = (C|T)$

$B$  has  $J$  independent columns; Rank  $J \times M$  matrix  $C$  is  $M$  and  
Rank  $J \times (J - M)$  matrix  $T$  is  $J - M$

Any  $\underline{w} \in J$  dimensional space spanned by columns  $B$ :

$\underline{w} = C \cdot \underline{v} - T \cdot \underline{w}_a$  with  $C \cdot \underline{v} = \underline{w}_c$  weights belong to constraint

$$\text{Constraint } \Rightarrow C^h \underline{w} = C^h C \underline{v} - C^h T \underline{w}_a = \underline{r}_d$$

Construct  $T$  such that  $C^h T = 0 \Rightarrow \underline{v} = (C^h C)^{-1} \underline{r}_d$

$$\underline{w}_c = C \cdot \underline{v} = \boxed{C \cdot (C^h C)^{-1} \underline{r}_d}$$

# Generalized Sidelobe Canceller

## Possible solutions for $\underline{w}_a$ :

With  $\underline{w} = \underline{w}_c - \mathbf{T} \cdot \underline{w}_a$  we can write constrained optimization:

$$\min_{\underline{w}} \{P_y\} = \min_{\underline{w}} \{\underline{w}^h \mathbf{R} \underline{w}\} \text{ s.t. } \mathbf{C}^h \cdot \underline{w} = \underline{r}_d$$

# Generalized Sidelobe Canceller

## Possible solutions for $\underline{w}_a$ :

With  $\underline{w} = \underline{w}_c - \mathbf{T} \cdot \underline{w}_a$  we can write constrained optimization:

$$\min_{\underline{w}} \{P_y\} = \min_{\underline{w}} \{\underline{w}^h \mathbf{R} \underline{w}\} \text{ s.t. } \mathbf{C}^h \cdot \underline{w} = \underline{r}_d$$

to following unconstrained optimization:

$$\min_{\underline{w}_a} \left\{ (\underline{w}_c - \mathbf{T} \underline{w}_a)^h \mathbf{R} (\underline{w}_c - \mathbf{T} \underline{w}_a) \right\}$$

# Generalized Sidelobe Canceller

Possible solutions for  $\underline{w}_a$ :

With  $\underline{w} = \underline{w}_c - T \cdot \underline{w}_a$  we can write constrained optimization:

$$\min_{\underline{w}} \{P_y\} = \min_{\underline{w}} \{\underline{w}^h R \underline{w}\} \text{ s.t. } C^h \cdot \underline{w} = \underline{r}_d$$

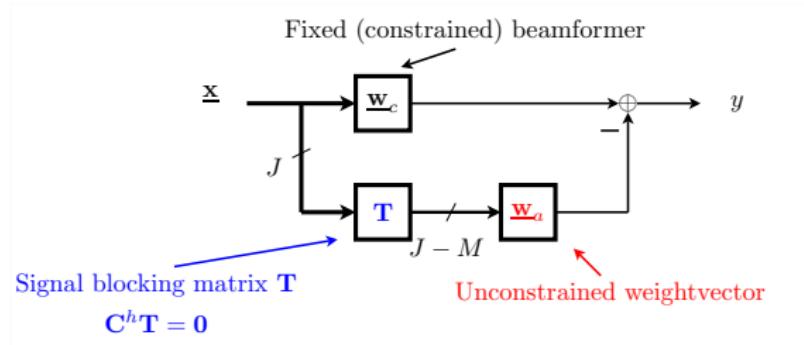
to following unconstrained optimization:

$$\min_{\underline{w}_a} \left\{ (\underline{w}_c - T \underline{w}_a)^h R (\underline{w}_c - T \underline{w}_a) \right\}$$

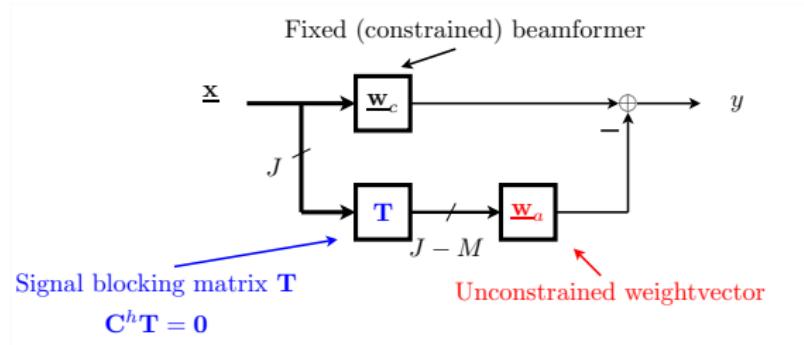
$$\frac{d}{d \underline{w}_a} = \underline{0} \Rightarrow -2T^h R \underline{w}_c + 2T^h R T \underline{w}_a = \underline{0} \Rightarrow$$

$$\underline{w}_a = (T^h R T)^{-1} T^h R \underline{w}_c$$

# Generalized Sidelobe Canceller



# Generalized Sidelobe Canceller



*Note:* Blocking matrix can be constructed by any orthogonalization procedure (e.g. Gramm- Schmidt or QR- decomposition)

# Generalized Sidelobe Canceller

**Example:**  $r_d = 1$  in steering direction  $\theta_0$

# Generalized Sidelobe Canceller

**Example:**  $r_d = 1$  in steering direction  $\theta_0$

$$C = \underline{a}(\omega_0, \theta_0) = (1, e^{-j\phi_0}, e^{-j2\phi_0}, \dots, e^{-j(J-1)\phi_0})^t$$

$$\text{with } \phi_0 = \omega_0 \frac{d \sin(\theta_0)}{c}$$

# Generalized Sidelobe Canceller

**Example:**  $r_d = 1$  in steering direction  $\theta_0$

$$C = \underline{a}(\omega_0, \theta_0) = (1, e^{-j\phi_0}, e^{-j2\phi_0}, \dots, e^{-j(J-1)\phi_0})^t$$

$$\text{with } \phi_0 = \omega_0 \frac{d \sin(\theta_0)}{c} \Rightarrow$$

$$\underline{w}_c = C \left( C^h C \right)^{-1} \underline{r}_d = \underline{a} \left( \underline{a}^h \underline{a} \right)^{-1} (1) = \frac{1}{J} \underline{a}(\omega_0, \theta_0)$$

# Generalized Sidelobe Canceller

**Example:**  $r_d = 1$  in steering direction  $\theta_0$

$$C = \underline{a}(\omega_0, \theta_0) = (1, e^{-j\phi_0}, e^{-j2\phi_0}, \dots, e^{-j(J-1)\phi_0})^t$$

$$\text{with } \phi_0 = \omega_0 \frac{d \sin(\theta_0)}{c} \Rightarrow$$

$$\underline{w}_c = C \left( C^h C \right)^{-1} \underline{r}_d = \underline{a} \left( \underline{a}^h \underline{a} \right)^{-1} (1) = \frac{1}{J} \underline{a}(\omega_0, \theta_0)$$

Beampattern of this fixed filter part:

$$|r(\omega, \theta)| = |\underline{w}_c^h \cdot \underline{a}(\omega, \theta)| = \left| \frac{1}{J} \sum_{i=1}^J e^{j(i-1) \frac{d}{c} (\omega_0 \sin(\theta_0) - \omega \sin(\theta))} \right|$$

# Generalized Sidelobe Canceller

**Example:**  $r_d = 1$  in steering direction  $\theta_0$

$$C = \underline{a}(\omega_0, \theta_0) = (1, e^{-j\phi_0}, e^{-j2\phi_0}, \dots, e^{-j(J-1)\phi_0})^t$$

$$\text{with } \phi_0 = \omega_0 \frac{d \sin(\theta_0)}{c} \Rightarrow$$

$$\underline{w}_c = C \left( C^h C \right)^{-1} r_d = \underline{a} \left( \underline{a}^h \underline{a} \right)^{-1} (1) = \frac{1}{J} \underline{a}(\omega_0, \theta_0)$$

Beampattern of this fixed filter part:

$$|r(\omega, \theta)| = |\underline{w}_c^h \cdot \underline{a}(\omega, \theta)| = \left| \frac{1}{J} \sum_{i=1}^J e^{j(i-1) \frac{d}{c} (\omega_0 \sin(\theta_0) - \omega \sin(\theta))} \right|$$

For  $\omega = \omega_0 = 2\pi f_0 = 2\pi \frac{c}{\lambda_0}$  beampattern becomes:

$$|r(\omega_0, \theta)| = \frac{1}{J} \frac{\sin \left( J \pi \frac{d}{\lambda_0} [\sin(\theta_0) - \sin(\theta)] \right)}{\sin \left( \pi \frac{d}{\lambda_0} [\sin(\theta_0) - \sin(\theta)] \right)}$$

# Generalized Sidelobe Canceller

Furthermore construction of blocking matrix:

$$C^h T = 0 \Rightarrow \underline{a}^h T = 0 \text{ with } J \times (J - 1) \text{ matrix } T$$

# Generalized Sidelobe Canceller

Furthermore construction of blocking matrix:

$$C^h T = 0 \Rightarrow \underline{a}^h T = 0 \text{ with } J \times (J-1) \text{ matrix } T$$

E.g.  $T = \begin{pmatrix} -1 & -1 & \dots & -1 \\ e^{-j\phi_0} & 0 & \dots & 0 \\ 0 & e^{-j2\cdot\phi_0} & \dots & 0 \\ 0 & 0 & \dots & e^{-j(J-1)\cdot\phi_0} \end{pmatrix}$

# Generalized Sidelobe Canceller

Furthermore construction of blocking matrix:

$$\mathbf{C}^h \mathbf{T} = 0 \Rightarrow \underline{\mathbf{a}}^h \mathbf{T} = 0 \text{ with } J \times (J-1) \text{ matrix } \mathbf{T}$$

E.g.  $\mathbf{T} = \begin{pmatrix} -1 & -1 & \dots & -1 \\ e^{-j\phi_0} & 0 & \dots & 0 \\ 0 & e^{-j2\cdot\phi_0} & \dots & 0 \\ 0 & 0 & \dots & e^{-j(J-1)\cdot\phi_0} \end{pmatrix}$

Thus for  $m^{th}$  column (with  $m = 1, 2, \dots, J-1$ )

$$\underline{\mathbf{t}}_m = \left( -1, 0, \dots, 0, e^{-j m \phi_0}, 0, \dots, 0 \right)^t$$

with beampattern  $|r_m| = |\underline{\mathbf{t}}_m^h \cdot \underline{\mathbf{a}}(\omega_0, \theta)|$

# Generalized Sidelobe Canceller

Furthermore construction of blocking matrix:

$$C^h T = 0 \Rightarrow \underline{a}^h T = 0 \text{ with } J \times (J-1) \text{ matrix } T$$

E.g.  $T = \begin{pmatrix} -1 & -1 & \dots & -1 \\ e^{-j\phi_0} & 0 & \dots & 0 \\ 0 & e^{-j2\cdot\phi_0} & \dots & 0 \\ 0 & 0 & \dots & e^{-j(J-1)\cdot\phi_0} \end{pmatrix}$

Thus for  $m^{th}$  column (with  $m = 1, 2, \dots, J-1$ )

$$\underline{t}_m = \left( -1, 0, \dots, 0, e^{-jm\phi_0}, 0, \dots, 0 \right)^t$$

with beampattern  $|r_m| = |\underline{t}_m^h \cdot \underline{a}(\omega_0, \theta)| \Rightarrow$  for  $d/\lambda = 1/2$ :

$$|r_m(\omega, \theta)| = 2 \left| \sin \left( \frac{1}{2} m \pi [\sin(\theta_0) - \sin(\theta)] \right) \right|$$

# Generalized Sidelobe Canceller

Furthermore construction of blocking matrix:

$$C^h T = 0 \Rightarrow \underline{a}^h T = 0 \text{ with } J \times (J-1) \text{ matrix } T$$

E.g.  $T = \begin{pmatrix} -1 & -1 & \dots & -1 \\ e^{-j\phi_0} & 0 & \dots & 0 \\ 0 & e^{-j2\cdot\phi_0} & \dots & 0 \\ 0 & 0 & \dots & e^{-j(J-1)\cdot\phi_0} \end{pmatrix}$

Thus for  $m^{th}$  column (with  $m = 1, 2, \dots, J-1$ )

$$\underline{t}_m = \left( -1, 0, \dots, 0, e^{-jm\phi_0}, 0, \dots, 0 \right)^t$$

with beampattern  $|r_m| = |\underline{t}_m^h \cdot \underline{a}(\omega_0, \theta)| \Rightarrow$  for  $d/\lambda = 1/2$ :

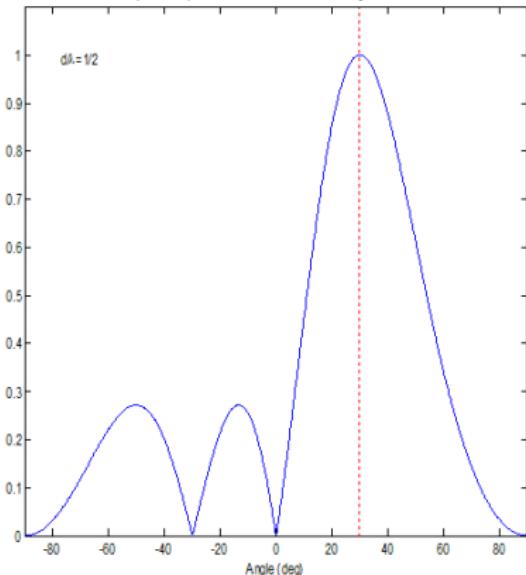
$$|r_m(\omega, \theta)| = 2 \left| \sin \left( \frac{1}{2} m \pi [\sin(\theta_0) - \sin(\theta)] \right) \right|$$

(= amplitude response of  $m^{th}$  column blocking matrix)

# Generalized Sidelobe Canceller

$$|r(\theta)|$$

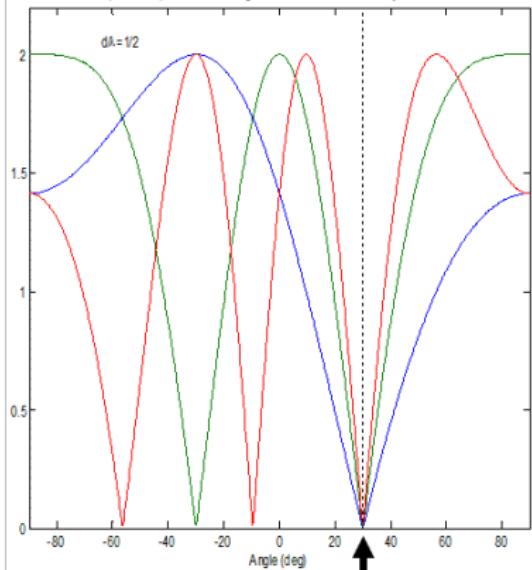
Amplitude response of fixed beamformer for array with 4 sensors



$\theta_{desired}$

$$|r_m(\theta)|$$

Amplitude responses of blocking matrix beamformers for array with 4 sensors



$\theta_{desired}$

# Adaptive Array Signal Processing

## Why adaptive?

In most previous results knowledge of SOS needed.

These statistics are usually unknown and/or time-varying

With ergodic assumption → SOS can be estimated from available data samples → adaptive solutions!

# Adaptive Array Signal Processing

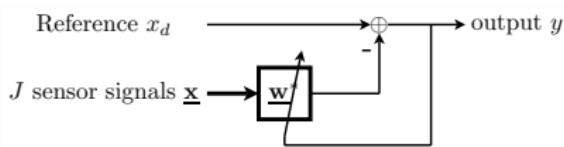
## Why adaptive?

In most previous results knowledge of SOS needed.

These statistics are usually unknown and/or time-varying

With ergodic assumption → SOS can be estimated from available data samples → adaptive solutions!

## General adaptive array structure:



$$\text{Wiener: } \underline{w}_{opt} = \arg \min_{\underline{w}} \{ E\{|y|^2 \} \}$$
$$\Rightarrow \quad \underline{w}_{opt} = \mathbf{R}_x^{-1} \cdot \underline{r}_{x x_d^*}$$

# Adaptive Array Signal Processing

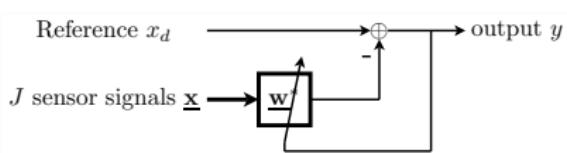
## Why adaptive?

In most previous results knowledge of SOS needed.

These statistics are usually unknown and/or time-varying

With ergodic assumption → SOS can be estimated from available data samples → adaptive solutions!

## General adaptive array structure:

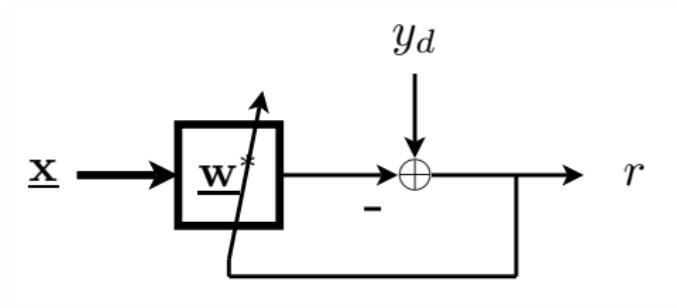


$$\text{Wiener: } \underline{\mathbf{w}}_{opt} = \arg \min_{\underline{\mathbf{w}}} \{ E\{|\mathbf{y}|^2\} \}$$
$$\Rightarrow \quad \underline{\mathbf{w}}_{opt} = \mathbf{R}_x^{-1} \cdot \mathbf{r}_{xx_d^*}$$

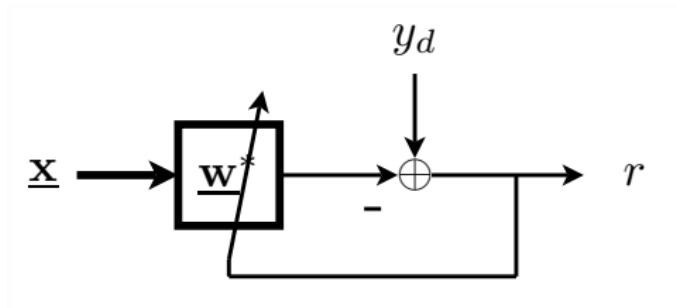
$$\text{LMS update rule : } \underline{\mathbf{w}}[k+1] = \underline{\mathbf{w}}[k] + 2\alpha \underline{\mathbf{x}}[k] y^*[k]$$

$$\text{Final value : } \lim_{k \rightarrow \infty} E\{\underline{\mathbf{w}}[k]\} = \underline{\mathbf{w}}_{opt}$$

## Adaptive MMSE (Wiener)



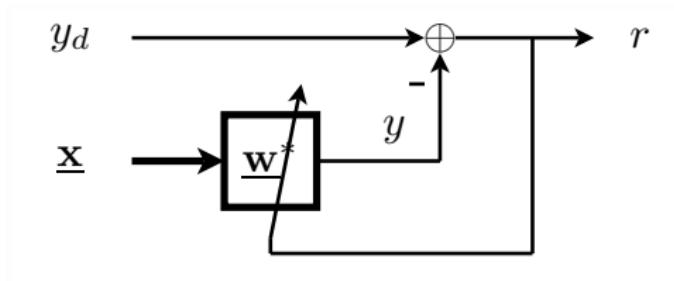
## Adaptive MMSE (Wiener)



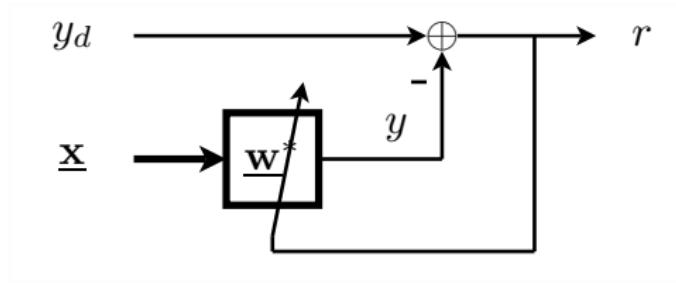
**Optimal Wiener** :  $\underline{w}_{opt} = \arg \min_{\underline{w}} \{ E\{|r|^2\} \} = R_x^{-1} \cdot r_{xy_d^*}$

**LMS update rule** :  $\underline{w}[k+1] = \underline{w}[k] + 2\alpha \underline{x}[k] r^*[k]$

# Adaptive MSC



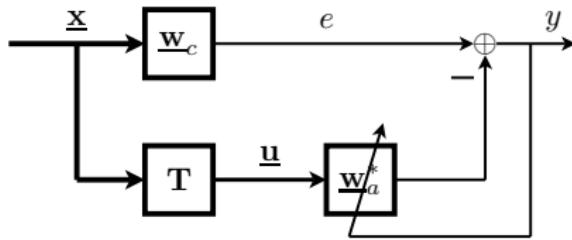
# Adaptive MSC



**Optimal Wiener** :  $\underline{w}_{opt} = \arg \min_{\underline{w}} \{E\{|r|^2\}\} = R_x^{-1} \cdot r_{xy_d^*}$

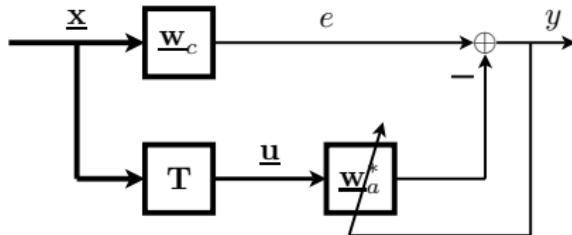
**LMS update rule** :  $\underline{w}[k+1] = \underline{w}[k] + 2\alpha \underline{x}[k] r^*[k]$

# Adaptive GSC



$$\begin{aligned}\underline{w}_c &= C \cdot (C^h C)^{-1} \underline{r}_d \\ C^h T &= 0 \\ \underline{u} &= T \cdot \underline{x}\end{aligned}$$

# Adaptive GSC



$$\begin{aligned}\underline{w}_c &= \underline{C} \cdot (\underline{C}^h \underline{C})^{-1} \underline{r}_d \\ \underline{C}^h \underline{T} &= 0 \\ \underline{u} &= \underline{T} \cdot \underline{x}\end{aligned}$$

**Optimal Wiener** :  $\underline{w}_{opt} = \arg \min_{\underline{w}} \{E\{|\underline{r}|^2\}\} = \underline{R}_u^{-1} \cdot \underline{r}_{ue*}$

**LMS update rule** :  $\underline{w}[k+1] = \underline{w}[k] + 2\alpha \underline{u}[k] y^*[k]$

# Appendix AASP

## Content appendix

- ▶ Eigenvalue problem
- ▶ Generalized inverse
- ▶ Projection matrix
- ▶ Matrix inversion lemma
- ▶ Signal subspace techniques

# Eigenvalue problem

Procedure: With eigenvalues  $\lambda_i$  and eigenvectors  $\underline{q}_i$ :

$$R \cdot \underline{q}_i = \lambda_i \cdot \underline{q}_i \Rightarrow (R - \lambda_i I) \cdot \underline{q}_i = \underline{0} \text{ for } i = 0, 1, \dots, N-1$$

# Eigenvalue problem

Procedure: With eigenvalues  $\lambda_i$  and eigenvectors  $\underline{q}_i$ :

$$R \cdot \underline{q}_i = \lambda_i \cdot \underline{q}_i \Rightarrow (R - \lambda_i I) \cdot \underline{q}_i = \underline{0} \text{ for } i = 0, 1, \dots, N-1$$

With  $Q = (\underline{q}_0, \dots, \underline{q}_{N-1})$  and  $\Lambda = \text{diag}\{\lambda_0, \dots, \lambda_{N-1}\}$

$$R \cdot Q = Q \cdot \Lambda \quad \Rightarrow \quad R = Q \Lambda Q^{-1}$$

# Eigenvalue problem

Procedure: With eigenvalues  $\lambda_i$  and eigenvectors  $\underline{q}_i$ :

$$R \cdot \underline{q}_i = \lambda_i \cdot \underline{q}_i \Rightarrow (R - \lambda_i I) \cdot \underline{q}_i = \underline{0} \text{ for } i = 0, 1, \dots, N-1$$

With  $Q = (\underline{q}_0, \dots, \underline{q}_{N-1})$  and  $\Lambda = \text{diag}\{\lambda_0, \dots, \lambda_{N-1}\}$

$$R \cdot Q = Q \cdot \Lambda \Rightarrow R = Q \Lambda Q^{-1}$$

Property: Eigenvectors  $\underline{q}_i$  orthogonal  $\Rightarrow$

$$Q^h \cdot Q = Q \cdot Q^h = c \cdot I \text{ with } c \text{ some constant}$$

# Eigenvalue problem

Procedure: With eigenvalues  $\lambda_i$  and eigenvectors  $\underline{q}_i$ :

$$R \cdot \underline{q}_i = \lambda_i \cdot \underline{q}_i \Rightarrow (R - \lambda_i I) \cdot \underline{q}_i = \underline{0} \text{ for } i = 0, 1, \dots, N-1$$

With  $Q = (\underline{q}_0, \dots, \underline{q}_{N-1})$  and  $\Lambda = \text{diag}\{\lambda_0, \dots, \lambda_{N-1}\}$

$$R \cdot Q = Q \cdot \Lambda \Rightarrow R = Q \Lambda Q^{-1}$$

Property: Eigenvectors  $\underline{q}_i$  orthogonal  $\Rightarrow$

$$Q^h \cdot Q = Q \cdot Q^h = c \cdot I \text{ with } c \text{ some constant}$$

Main result:

**Diagonalization:** 
$$Q^h R Q = \Lambda \Leftrightarrow R = Q \Lambda Q^h$$

# Eigenvalue problem

Example MA(1):

$$x[k] = i[k] + ai[k - 1] \text{ with } E\{i[k]\} = 0 \text{ and } E\{i^2[k]\} = \sigma_i^2 \Rightarrow$$

# Eigenvalue problem

## Example MA(1):

$x[k] = i[k] + ai[k - 1]$  with  $E\{i[k]\} = 0$  and  $E\{i^2[k]\} = \sigma_i^2 \Rightarrow$

$\rho[0] = (1 + a^2)\sigma_i^2; \rho[1] = \rho[-1] = a\sigma_i^2; \rho[\tau] = 0$  for  $|\tau| \geq 2$

# Eigenvalue problem

## Example MA(1):

$x[k] = i[k] + ai[k - 1]$  with  $E\{i[k]\} = 0$  and  $E\{i^2[k]\} = \sigma_i^2 \Rightarrow$

$\rho[0] = (1 + a^2)\sigma_i^2; \rho[1] = \rho[-1] = a\sigma_i^2; \rho[\tau] = 0$  for  $|\tau| \geq 2$

Eigenvalues problem  $\det(R - \lambda I) = 0$  for  $N = 2$  (with  $\gamma = \rho[1]/\rho[0]$ ):

$$\Lambda = \begin{pmatrix} \lambda_0 & 0 \\ 0 & \lambda_1 \end{pmatrix} = \begin{pmatrix} 1 + \gamma & 0 \\ 0 & 1 - \gamma \end{pmatrix} ; Q = (\underline{q}_0, \underline{q}_1) = c \cdot \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix}$$

# Eigenvalue problem

## Example MA(1):

$x[k] = i[k] + ai[k - 1]$  with  $E\{i[k]\} = 0$  and  $E\{i^2[k]\} = \sigma_i^2 \Rightarrow$

$\rho[0] = (1 + a^2)\sigma_i^2; \rho[1] = \rho[-1] = a\sigma_i^2; \rho[\tau] = 0$  for  $|\tau| \geq 2$

Eigenvalues problem  $\det(R - \lambda I) = 0$  for  $N = 2$  (with  $\gamma = \rho[1]/\rho[0]$ ):

$$\Lambda = \begin{pmatrix} \lambda_0 & 0 \\ 0 & \lambda_1 \end{pmatrix} = \begin{pmatrix} 1 + \gamma & 0 \\ 0 & 1 - \gamma \end{pmatrix}; Q = (\underline{q}_0, \underline{q}_1) = c \cdot \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix}$$

Notes:

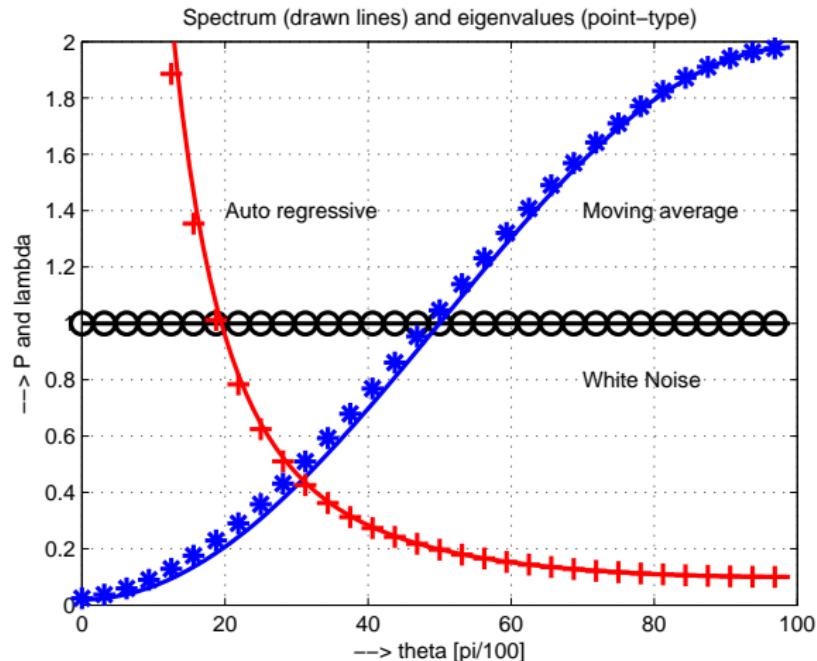
- ▶ Vector  $\underline{q}_0$  orthogonal to  $\underline{q}_1$  since  $\underline{q}_0^t \cdot \underline{q}_1 = 0$
- ▶ For white noise ( $a = 0$ ):  $\Lambda = I$
- ▶ For MA(1) with  $N > 2$ :  $R$  is tri-diagonal

# Eigenvalue problem

Example: Eigenvalues and psd for white noise, MA(1) and AR(1)

# Eigenvalue problem

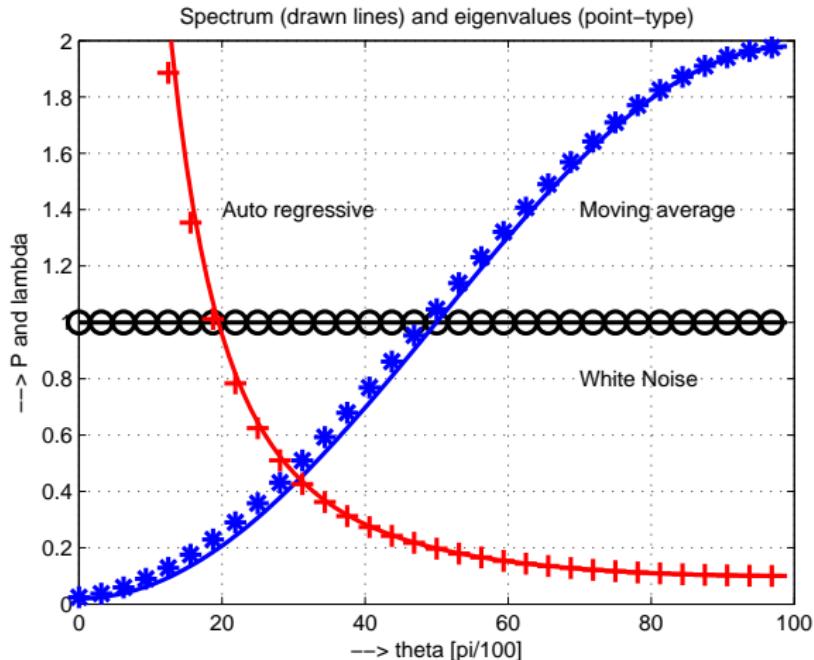
Example: Eigenvalues and psd for white noise, MA(1) and AR(1)



# Eigenvalue problem

Back2Slides

Example: Eigenvalues and psd for white noise, MA(1) and AR(1)



# Generalized inverse

## Goal:

For known  $M \times N$  matrix  $A$  and  $M \times 1$  vector  $\underline{b}$ , solve linear set of  $M$  equations and  $N$  unknowns:

$$A \cdot \underline{w} = \underline{b}$$

# Generalized inverse

## Goal:

For known  $M \times N$  matrix  $A$  and  $M \times 1$  vector  $\underline{b}$ , solve linear set of  $M$  equations and  $N$  unknowns:

$$A \cdot \underline{w} = \underline{b}$$

General solution for  $N \times 1$  vector  $\underline{w}$ :

$$\underline{w} = A^\dagger \cdot \underline{b}$$

with  $\dagger$  the generalized (Moore-Penrose) pseudo inverse, defined as:

# Generalized inverse

## Goal:

For known  $M \times N$  matrix  $A$  and  $M \times 1$  vector  $\underline{b}$ , solve linear set of  $M$  equations and  $N$  unknowns:

$$A \cdot \underline{w} = \underline{b}$$

General solution for  $N \times 1$  vector  $\underline{w}$ :

$$\underline{w} = A^\dagger \cdot \underline{b}$$

with  $\dagger$  the generalized (Moore-Penrose) pseudo inverse, defined as:

$$A^\dagger = A^h \left( A A^h \right)^{-1} \quad \text{for } M < N$$

$$A^\dagger = A^{-1} \quad \text{for } M = N$$

$$A^\dagger = \left( A^h A \right)^{-1} A^h \quad \text{for } M > N$$

## Generalized inverse

**Case  $M < N$ :**  $\underline{w} = A^\dagger \cdot \underline{b} \Rightarrow$  Multiple solutions

## Generalized inverse

**Case  $M < N$ :**  $\underline{w} = A^\dagger \cdot \underline{b} \Rightarrow$  Multiple solutions

Example  $M = 1, N = 2$ :

$$w_1 + w_2 = 2 \Rightarrow A \cdot \underline{w} = \underline{b} \Leftrightarrow (1, 1) \cdot \begin{pmatrix} w_1 \\ w_2 \end{pmatrix} = (2)$$

## Generalized inverse

**Case  $M < N$ :**  $\underline{w} = A^\dagger \cdot \underline{b} \Rightarrow$  Multiple solutions

Example  $M = 1, N = 2$ :

$$w_1 + w_2 = 2 \Rightarrow A \cdot \underline{w} = \underline{b} \Leftrightarrow (1, 1) \cdot \begin{pmatrix} w_1 \\ w_2 \end{pmatrix} = (2)$$

$$\underline{w} = A^\dagger \cdot \underline{b} = \begin{pmatrix} 1 \\ 1 \end{pmatrix}$$

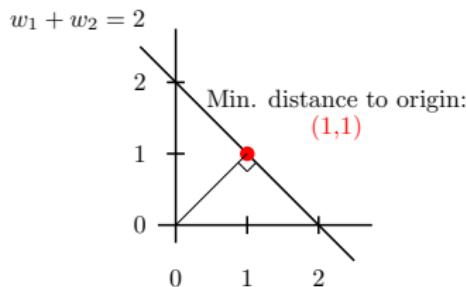
## Generalized inverse

**Case  $M < N$ :**  $\underline{w} = A^\dagger \cdot \underline{b} \Rightarrow$  Multiple solutions

Example  $M = 1, N = 2$ :

$$w_1 + w_2 = 2 \Rightarrow A \cdot \underline{w} = \underline{b} \Leftrightarrow (1, 1) \cdot \begin{pmatrix} w_1 \\ w_2 \end{pmatrix} = (2)$$

$$\underline{w} = A^\dagger \cdot \underline{b} = \begin{pmatrix} 1 \\ 1 \end{pmatrix}$$



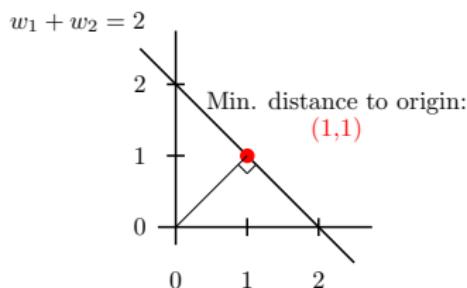
# Generalized inverse

**Case  $M < N$ :**  $\underline{w} = A^\dagger \cdot \underline{b} \Rightarrow$  Multiple solutions

Example  $M = 1, N = 2$ :

$$w_1 + w_2 = 2 \Rightarrow A \cdot \underline{w} = \underline{b} \Leftrightarrow (1, 1) \cdot \begin{pmatrix} w_1 \\ w_2 \end{pmatrix} = (2)$$

$$\underline{w} = A^\dagger \cdot \underline{b} = \begin{pmatrix} 1 \\ 1 \end{pmatrix}$$



## Conclusion:

$\dagger$  results in solution with smallest Euclidian norm ("minimum distance to the origin (0,0)")

## Generalized inverse

**Case  $M = N$ :**  $\underline{w} = A^{-1} \cdot \underline{b}$  ( $A$  must be invertable)

## Generalized inverse

**Case  $M = N$ :**  $\underline{w} = A^{-1} \cdot \underline{b}$  ( $A$  must be invertable)

Example  $M = 2, N = 2$ :

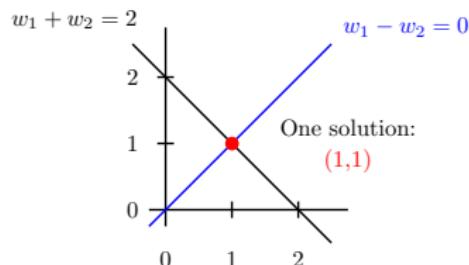
$$\begin{pmatrix} 1 & -1 \\ 1 & 1 \end{pmatrix} \cdot \begin{pmatrix} w_1 \\ w_2 \end{pmatrix} = \begin{pmatrix} 0 \\ 2 \end{pmatrix} \Rightarrow \begin{pmatrix} w_1 \\ w_2 \end{pmatrix} = \begin{pmatrix} 1 \\ 1 \end{pmatrix}$$

# Generalized inverse

**Case  $M = N$ :**  $\underline{w} = A^{-1} \cdot \underline{b}$  ( $A$  must be invertable)

Example  $M = 2, N = 2$ :

$$\begin{pmatrix} 1 & -1 \\ 1 & 1 \end{pmatrix} \cdot \begin{pmatrix} w_1 \\ w_2 \end{pmatrix} = \begin{pmatrix} 0 \\ 2 \end{pmatrix} \Rightarrow \begin{pmatrix} w_1 \\ w_2 \end{pmatrix} = \begin{pmatrix} 1 \\ 1 \end{pmatrix}$$

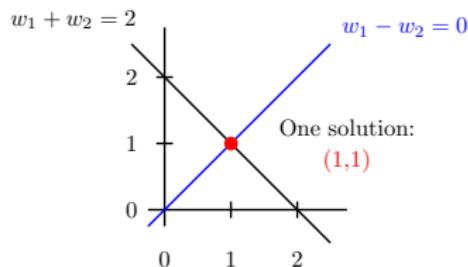


## Generalized inverse

**Case  $M = N$ :**  $\underline{w} = A^{-1} \cdot \underline{b}$  ( $A$  must be invertable)

Example  $M = 2, N = 2$ :

$$\begin{pmatrix} 1 & -1 \\ 1 & 1 \end{pmatrix} \cdot \begin{pmatrix} w_1 \\ w_2 \end{pmatrix} = \begin{pmatrix} 0 \\ 2 \end{pmatrix} \Rightarrow \begin{pmatrix} w_1 \\ w_2 \end{pmatrix} = \begin{pmatrix} 1 \\ 1 \end{pmatrix}$$



**Conclusion:**  $\dagger$  results in unique solution

## Generalized inverse

**Case  $M > N$ :** Overdetermined set of equations

## Generalized inverse

**Case  $M > N$ :** Overdetermined set of equations

**Min. norm solution:**  $\underline{w} = \arg \min_{\underline{w}} \|\underline{A}\underline{w} - \underline{b}\|^2$

## Generalized inverse

**Case  $M > N$ :** Overdetermined set of equations

**Min. norm solution:**  $\underline{w} = \arg \min_{\underline{w}} \|\underline{A}\underline{w} - \underline{b}\|^2$

$$J = \|\underline{A}\underline{w} - \underline{b}\|^2 = \underline{w}^h \underline{A}^h \underline{A}\underline{w} - \underline{w}^h \underline{A}^h \underline{b} - \underline{b}^h \underline{A}\underline{w} + \underline{b}^h \underline{b}$$

## Generalized inverse

**Case  $M > N$ :** Overdetermined set of equations

**Min. norm solution:**  $\underline{w} = \arg \min_{\underline{w}} \|\underline{A}\underline{w} - \underline{b}\|^2$

$$J = \|\underline{A}\underline{w} - \underline{b}\|^2 = \underline{w}^h \underline{A}^h \underline{A}\underline{w} - \underline{w}^h \underline{A}^h \underline{b} - \underline{b}^h \underline{A}\underline{w} + \underline{b}^h \underline{b}$$

$$\Rightarrow \frac{dJ}{d\underline{w}} = 0 \Rightarrow \underline{w} = (\underline{A}^h \underline{A})^{-1} \underline{A}^h \cdot \underline{b} = \underline{A}^\dagger \cdot \underline{b}$$

## Generalized inverse

**Case  $M > N$ :** Overdetermined set of equations

**Min. norm solution:**  $\underline{w} = \arg \min_{\underline{w}} \|\underline{A}\underline{w} - \underline{b}\|^2$

$$J = \|\underline{A}\underline{w} - \underline{b}\|^2 = \underline{w}^h \underline{A}^h \underline{A}\underline{w} - \underline{w}^h \underline{A}^h \underline{b} - \underline{b}^h \underline{A}\underline{w} + \underline{b}^h \underline{b}$$

$$\Rightarrow \frac{dJ}{d\underline{w}} = 0 \Rightarrow \underline{w} = (\underline{A}^h \underline{A})^{-1} \underline{A}^h \cdot \underline{b} = \underline{A}^\dagger \cdot \underline{b}$$

Example  $M = 3, N = 2$ :

$$\begin{pmatrix} 1 & -1 \\ 1 & 1 \\ 1 & 0 \end{pmatrix} \cdot \begin{pmatrix} w_1 \\ w_2 \end{pmatrix} = \begin{pmatrix} 0 \\ 2 \\ 2 \end{pmatrix} \Rightarrow \underline{w} = \underline{A}^\dagger \cdot \underline{b} = \begin{pmatrix} \frac{4}{3} \\ 1 \end{pmatrix}$$

# Generalized inverse

**Case  $M > N$ :** Overdetermined set of equations

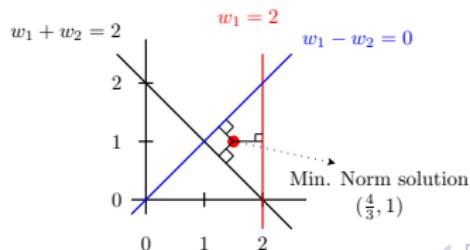
**Min. norm solution:**  $\underline{w} = \arg \min_{\underline{w}} \|\underline{A}\underline{w} - \underline{b}\|^2$

$$J = \|\underline{A}\underline{w} - \underline{b}\|^2 = \underline{w}^h \underline{A}^h \underline{A}\underline{w} - \underline{w}^h \underline{A}^h \underline{b} - \underline{b}^h \underline{A}\underline{w} + \underline{b}^h \underline{b}$$

$$\Rightarrow \frac{dJ}{d\underline{w}} = 0 \Rightarrow \underline{w} = (\underline{A}^h \underline{A})^{-1} \underline{A}^h \cdot \underline{b} = \underline{A}^\dagger \cdot \underline{b}$$

Example  $M = 3, N = 2$ :

$$\begin{pmatrix} 1 & -1 \\ 1 & 1 \\ 1 & 0 \end{pmatrix} \cdot \begin{pmatrix} w_1 \\ w_2 \end{pmatrix} = \begin{pmatrix} 0 \\ 2 \\ 2 \end{pmatrix} \Rightarrow \underline{w} = \underline{A}^\dagger \cdot \underline{b} = \begin{pmatrix} \frac{4}{3} \\ 1 \end{pmatrix}$$



# Generalized inverse

Back2Slides

**Case  $M > N$ :** Overdetermined set of equations

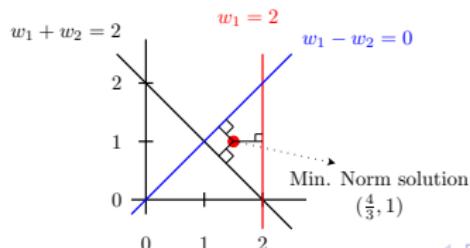
**Min. norm solution:**  $\underline{w} = \arg \min_{\underline{w}} \|\underline{A}\underline{w} - \underline{b}\|^2$

$$J = \|\underline{A}\underline{w} - \underline{b}\|^2 = \underline{w}^h \underline{A}^h \underline{A} \underline{w} - \underline{w}^h \underline{A}^h \underline{b} - \underline{b}^h \underline{A} \underline{w} + \underline{b}^h \underline{b}$$

$$\Rightarrow \frac{dJ}{d\underline{w}} = 0 \Rightarrow \underline{w} = (\underline{A}^h \underline{A})^{-1} \underline{A}^h \cdot \underline{b} = \underline{A}^\dagger \cdot \underline{b}$$

Example  $M = 3, N = 2$ :

$$\begin{pmatrix} 1 & -1 \\ 1 & 1 \\ 1 & 0 \end{pmatrix} \cdot \begin{pmatrix} w_1 \\ w_2 \end{pmatrix} = \begin{pmatrix} 0 \\ 2 \\ 2 \end{pmatrix} \Rightarrow \underline{w} = \underline{A}^\dagger \cdot \underline{b} = \begin{pmatrix} \frac{4}{3} \\ 1 \end{pmatrix}$$



# Projection matrix

Square matrix  $P$  is **Projection** matrix if:  $P^2 = P$

# Projection matrix

Square matrix  $P$  is **Projection** matrix if:  $P^2 = P$

**Orthogonal** projection matrix:  $P^h = P$  and  $P^2 = P$

# Projection matrix

Square matrix  $P$  is **Projection** matrix if:  $P^2 = P$

**Orthogonal** projection matrix:  $P^h = P$  and  $P^2 = P$

**General:**  $N \times M$  matrix  $V$ , with linearly independent columns

Projection ( $N \times 1$ )  $\underline{b}$  onto  $M$ -dim subspace spanned by columns  $V$ :

$$\hat{\underline{b}} = P_V \cdot \underline{b} \quad \text{and} \quad \underline{b}^\perp = (I - P_V) \cdot \underline{b}$$

with projection matrix :

$$P_V = V \left( V^h V \right)^{-1} V^h$$

# Projection matrix

Square matrix  $P$  is **Projection** matrix if:  $P^2 = P$

**Orthogonal** projection matrix:  $P^h = P$  and  $P^2 = P$

**General:**  $N \times M$  matrix  $V$ , with linearly independent columns

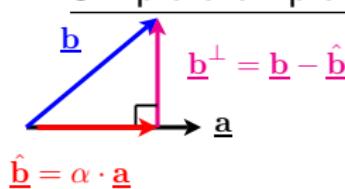
Projection ( $N \times 1$ )  $\underline{b}$  onto  $M$ -dim subspace spanned by columns  $V$ :

$$\hat{\underline{b}} = P_V \cdot \underline{b} \quad \text{and} \quad \underline{b}^\perp = (I - P_V) \cdot \underline{b}$$

with projection matrix :

$$P_V = V \left( V^h V \right)^{-1} V^h$$

Simple example:



$$\begin{aligned}\hat{\underline{b}}^h \cdot \underline{b}^\perp &= 0 \Rightarrow \alpha = (\underline{a}^h \underline{a})^{-1} \underline{a}^h \cdot \underline{b} \\ \Rightarrow \hat{\underline{b}} &= P_a \cdot \underline{b} \quad \text{and} \quad \underline{b}^\perp = (I - P_a) \cdot \underline{b} \\ \text{with } P_a &= \underline{a} (\underline{a}^h \underline{a})^{-1} \underline{a}^h\end{aligned}$$

# Matrix inversion lemma

Matrix dimensions: A:  $N \times N$ ; B:  $N \times M$ ; C:  $M \times M$ ; D:  $M \times N$

$$(A + BCD)^{-1} = A^{-1} - A^{-1}B(DA^{-1}B + C^{-1})^{-1}DA^{-1}$$

# Matrix inversion lemma

Matrix dimensions: A:  $N \times N$ ; B:  $N \times M$ ; C:  $M \times M$ ; D:  $M \times N$

$$(A + BCD)^{-1} = A^{-1} - A^{-1}B(DA^{-1}B + C^{-1})^{-1}DA^{-1}$$

Simple example: (scalar case)

$$(a + x \cdot y)^{-1} = \frac{1}{a + x \cdot y} = \dots = a^{-1} - \frac{a^{-1}xya^{-1}}{1 + ya^{-1}x}$$

# Matrix inversion lemma

Matrix dimensions: A:  $N \times N$ ; B:  $N \times M$ ; C:  $M \times M$ ; D:  $M \times N$

$$(A + BCD)^{-1} = A^{-1} - A^{-1}B(DA^{-1}B + C^{-1})^{-1}DA^{-1}$$

Simple example: (scalar case)

$$(a + x \cdot y)^{-1} = \frac{1}{a + x \cdot y} = \dots = a^{-1} - \frac{a^{-1}xya^{-1}}{1 + ya^{-1}x}$$

Special case: (RLS-like)

$B = \underline{x}$ :  $N \times 1$ ;  $C = 1$ :  $1 \times 1$  and  $D = \underline{x}^h$

$$(A + \underline{x}\underline{x}^h)^{-1} = A^{-1} - \frac{A^{-1}\underline{x}\underline{x}^hA^{-1}}{1 + \underline{x}^hA^{-1}\underline{x}}$$

# Signal subspace techniques

**Goal:** Determine spectral peaks in noisy measurements

# Signal subspace techniques

Goal: Determine spectral peaks in noisy measurements

Signal model:  $J$  sensors,  $P$  sources ( $P < J$ ). For  $i = 1, \dots, J$ :

$$x_i = \sum_{p=1}^P a_i(\theta_p) \cdot s_p[k] + n_i[k] \Leftrightarrow \underline{x}[k] = \mathbf{A} \cdot \underline{s}[k] + \underline{n}[k]$$

# Signal subspace techniques

Goal: Determine spectral peaks in noisy measurements

Signal model:  $J$  sensors,  $P$  sources ( $P < J$ ). For  $i = 1, \dots, J$ :

$$x_i = \sum_{p=1}^P a_i(\theta_p) \cdot s_p[k] + n_i[k] \Leftrightarrow \underline{x}[k] = \mathbf{A} \cdot \underline{s}[k] + \underline{n}[k]$$

Covariance structure:

$$\mathbf{R}_x = E\{\underline{x} \cdot \underline{x}^h\} = \mathbf{A} \mathbf{R}_s \mathbf{A}^h + \mathbf{R}_n$$

with  $\mathbf{R}_s = E\{\underline{s}\underline{s}^h\} = \text{diag}\{\sigma_{s_1}^2, \dots, \sigma_{s_P}^2\}$  ;  $\mathbf{R}_n = \sigma_n^2 \mathbf{I}$

and  $J \times P$  steering matrix  $\mathbf{A} = (\underline{a}(\theta_1), \dots, \underline{a}(\theta_P))$

# Signal subspace techniques

**What about rank of these matrices?**

# Signal subspace techniques

**What about rank of these matrices?**

$$\text{Rank}\{\mathbf{A}\} = P; \text{Rank}\{\mathbf{R}_s\} = P \Rightarrow \text{Rank}\{\mathbf{A}\mathbf{R}_s\mathbf{A}^h\} = P$$

# Signal subspace techniques

**What about rank of these matrices?**

$$\text{Rank}\{\mathbf{A}\} = P; \text{Rank}\{\mathbf{R}_s\} = P \Rightarrow \text{Rank}\{\mathbf{A}\mathbf{R}_s\mathbf{A}^h\} = P$$

$$\text{Furthermore since } \text{Rank}\{\mathbf{R}_n\} = J \Rightarrow \text{Rank}\{\mathbf{R}_x\} = J$$

# Signal subspace techniques

What about rank of these matrices?

$$\text{Rank}\{\mathbf{A}\} = P; \text{Rank}\{\mathbf{R}_s\} = P \Rightarrow \text{Rank}\{\mathbf{A}\mathbf{R}_s\mathbf{A}^h\} = P$$

$$\text{Furthermore since } \text{Rank}\{\mathbf{R}_n\} = J \Rightarrow \text{Rank}\{\mathbf{R}_x\} = J$$

Eigenvalue decomposition source signal part:

$$\left(\mathbf{A}\mathbf{R}_s\mathbf{A}^h\right) \cdot \underline{\mathbf{u}}_i = \lambda_{s_i} \cdot \underline{\mathbf{u}}_i \quad i = 1, \dots, J$$

# Signal subspace techniques

## What about rank of these matrices?

$$\text{Rank}\{\mathbf{A}\} = P; \text{Rank}\{\mathbf{R}_s\} = P \Rightarrow \text{Rank}\{\mathbf{A}\mathbf{R}_s\mathbf{A}^h\} = P$$

$$\text{Furthermore since } \text{Rank}\{\mathbf{R}_n\} = J \Rightarrow \text{Rank}\{\mathbf{R}_x\} = J$$

Eigenvalue decomposition source signal part:

$$\left(\mathbf{A}\mathbf{R}_s\mathbf{A}^h\right) \cdot \underline{\mathbf{u}}_i = \lambda_{s_i} \cdot \underline{\mathbf{u}}_i \quad i = 1, \dots, J$$

$$\text{Since } \text{Rank}\{\mathbf{A}\mathbf{R}_s\mathbf{A}^h\} = P \Rightarrow \lambda_{s_{P+1}} = \dots = \lambda_{s_J} = 0$$

# Signal subspace techniques

## What about rank of these matrices?

$$\text{Rank}\{\mathbf{A}\} = P; \text{Rank}\{\mathbf{R}_s\} = P \Rightarrow \text{Rank}\{\mathbf{A}\mathbf{R}_s\mathbf{A}^h\} = P$$

$$\text{Furthermore since } \text{Rank}\{\mathbf{R}_n\} = J \Rightarrow \text{Rank}\{\mathbf{R}_x\} = J$$

Eigenvalue decomposition source signal part:

$$\left(\mathbf{A}\mathbf{R}_s\mathbf{A}^h\right) \cdot \underline{\mathbf{u}}_i = \lambda_{s_i} \cdot \underline{\mathbf{u}}_i \quad i = 1, \dots, J$$

$$\text{Since } \text{Rank}\{\mathbf{A}\mathbf{R}_s\mathbf{A}^h\} = P \Rightarrow \lambda_{s_{P+1}} = \dots = \lambda_{s_J} = 0$$

Order eigenvalues:  $\lambda_{s_1} \geq \dots \geq \lambda_{s_P} > 0$

# Signal subspace techniques

## What about rank of these matrices?

$$\text{Rank}\{\mathbf{A}\} = P; \text{Rank}\{\mathbf{R}_s\} = P \Rightarrow \text{Rank}\{\mathbf{A}\mathbf{R}_s\mathbf{A}^h\} = P$$

$$\text{Furthermore since } \text{Rank}\{\mathbf{R}_n\} = J \Rightarrow \text{Rank}\{\mathbf{R}_x\} = J$$

Eigenvalue decomposition source signal part:

$$\left(\mathbf{A}\mathbf{R}_s\mathbf{A}^h\right) \cdot \underline{\mathbf{u}}_i = \lambda_{s_i} \cdot \underline{\mathbf{u}}_i \quad i = 1, \dots, J$$

$$\text{Since } \text{Rank}\{\mathbf{A}\mathbf{R}_s\mathbf{A}^h\} = P \Rightarrow \lambda_{s_{P+1}} = \dots = \lambda_{s_J} = 0$$

Order eigenvalues:  $\lambda_{s_1} \geq \dots \geq \lambda_{s_P} > 0$

$$\Rightarrow \left(\mathbf{A}\mathbf{R}_s\mathbf{A}^h\right) \cdot \mathbf{U}_s = \mathbf{U}_s \cdot \Lambda_s$$

with  $\mathbf{U}_s = (\underline{\mathbf{u}}_1, \dots, \underline{\mathbf{u}}_P)$ ;  $\Lambda_s = \text{diag}\{\lambda_{s_1}, \dots, \lambda_{s_P}\}$

# Signal subspace techniques

Eigenvalue decomposition input signal (use  $\underline{u}_i$  for  $i = 1, \dots, J$ ):

$$\mathbf{R}_x \cdot \underline{u}_i = (\mathbf{A}\mathbf{R}_s\mathbf{A}^h) \cdot \underline{u}_i + \sigma_n^2 \mathbf{I} \cdot \underline{u}_i = (\lambda_{s_i} + \sigma_n^2) \cdot \underline{u}_i$$

## Signal subspace techniques

Eigenvalue decomposition input signal (use  $\underline{u}_i$  for  $i = 1, \dots, J$ ):

$$\mathbf{R}_x \cdot \underline{u}_i = (\mathbf{A}\mathbf{R}_s\mathbf{A}^h) \cdot \underline{u}_i + \sigma_n^2 \mathbf{I} \cdot \underline{u}_i = (\lambda_{s_i} + \sigma_n^2) \cdot \underline{u}_i$$

Thus eigenvalues can be divided into two groups:

$$\lambda_{x_i} = \begin{cases} \lambda_{s_i} + \sigma_n^2 & \text{for } i = 1, \dots, P \\ \sigma_n^2 & \text{for } i = P + 1, \dots, J \end{cases}$$

# Signal subspace techniques

Eigenvalue decomposition input signal (use  $\underline{u}_i$  for  $i = 1, \dots, J$ ):

$$\mathbf{R}_x \cdot \underline{u}_i = (\mathbf{A}\mathbf{R}_s\mathbf{A}^h) \cdot \underline{u}_i + \sigma_n^2 \mathbf{I} \cdot \underline{u}_i = (\lambda_{s_i} + \sigma_n^2) \cdot \underline{u}_i$$

Thus eigenvalues can be divided into two groups:

$$\lambda_{x_i} = \begin{cases} \lambda_{s_i} + \sigma_n^2 & \text{for } i = 1, \dots, P \\ \sigma_n^2 & \text{for } i = P + 1, \dots, J \end{cases}$$

With  $\mathbf{U}_x = (\underline{u}_1, \dots, \underline{u}_J)$  and  $\Lambda_x = \text{diag}\{\lambda_{x_1}, \dots, \lambda_{x_J}\}$  we can write:

$$\begin{aligned} \mathbf{R}_x &= \mathbf{U}_x \Lambda_x \mathbf{U}_x^h = \sum_{i=1}^J \lambda_{x_i} \underline{u}_i \underline{u}_i^h = \sum_{i=1}^P (\lambda_{s_i} + \sigma_n^2) \underline{u}_i \underline{u}_i^h + \sum_{i=P+1}^J \sigma_n^2 \underline{u}_i \underline{u}_i^h \\ &= \mathbf{U}_s \Lambda_{s,n} \mathbf{U}_s^h + \mathbf{U}_n \Lambda_n \mathbf{U}_n^h \end{aligned}$$

# Signal subspace techniques

- Signal subspace** :  $U_s = (\underline{u}_1, \dots, \underline{u}_P)$   
 $\Lambda_{s,n} = \text{diag}\{\lambda_{s_1} + \sigma_n^2, \dots, \lambda_{s_P} + \sigma_n^2\}$
- Noise subspace** :  $U_n = (\underline{u}_{P+1}, \dots, \underline{u}_J)$   
 $\Lambda_n = \text{diag}\{\sigma_n^2, \dots, \sigma_n^2\}$

# Signal subspace techniques

**Signal subspace** :  $\mathbf{U}_s = (\underline{\mathbf{u}}_1, \dots, \underline{\mathbf{u}}_P)$

$$\Lambda_{s,n} = \text{diag}\{\lambda_{s1} + \sigma_n^2, \dots, \lambda_{sP} + \sigma_n^2\}$$

**Noise subspace** :  $\mathbf{U}_n = (\underline{\mathbf{u}}_{P+1}, \dots, \underline{\mathbf{u}}_J)$

$$\Lambda_n = \text{diag}\{\sigma_n^2, \dots, \sigma_n^2\}$$

Since  $\mathbf{U}_x = (\underline{\mathbf{u}}_1, \dots, \underline{\mathbf{u}}_J) = (\mathbf{U}_s, \mathbf{U}_n)$

and all eigenvectors orthogonal  $\underline{\mathbf{u}}_i \perp \underline{\mathbf{u}}_j$

$$\Rightarrow \boxed{\mathbf{U}_s \perp \mathbf{U}_n \Leftrightarrow \mathbf{U}_s^h \cdot \mathbf{U}_n = 0 \Leftrightarrow \mathbf{U}_n^h \cdot \mathbf{U}_s = 0}$$

# Signal subspace techniques

**Signal subspace** :  $\mathbf{U}_s = (\underline{\mathbf{u}}_1, \dots, \underline{\mathbf{u}}_P)$

$$\Lambda_{s,n} = \text{diag}\{\lambda_{s1} + \sigma_n^2, \dots, \lambda_{sP} + \sigma_n^2\}$$

**Noise subspace** :  $\mathbf{U}_n = (\underline{\mathbf{u}}_{P+1}, \dots, \underline{\mathbf{u}}_J)$

$$\Lambda_n = \text{diag}\{\sigma_n^2, \dots, \sigma_n^2\}$$

Since  $\mathbf{U}_x = (\underline{\mathbf{u}}_1, \dots, \underline{\mathbf{u}}_J) = (\mathbf{U}_s, \mathbf{U}_n)$

and all eigenvectors orthogonal  $\underline{\mathbf{u}}_i \perp \underline{\mathbf{u}}_j$

$$\Rightarrow \boxed{\mathbf{U}_s \perp \mathbf{U}_n \Leftrightarrow \mathbf{U}_s^h \cdot \mathbf{U}_n = 0 \Leftrightarrow \mathbf{U}_n^h \cdot \mathbf{U}_s = 0}$$

## Conclusion:

Any vector from signal subspace is orthogonal to noise subspace

End

Appendix