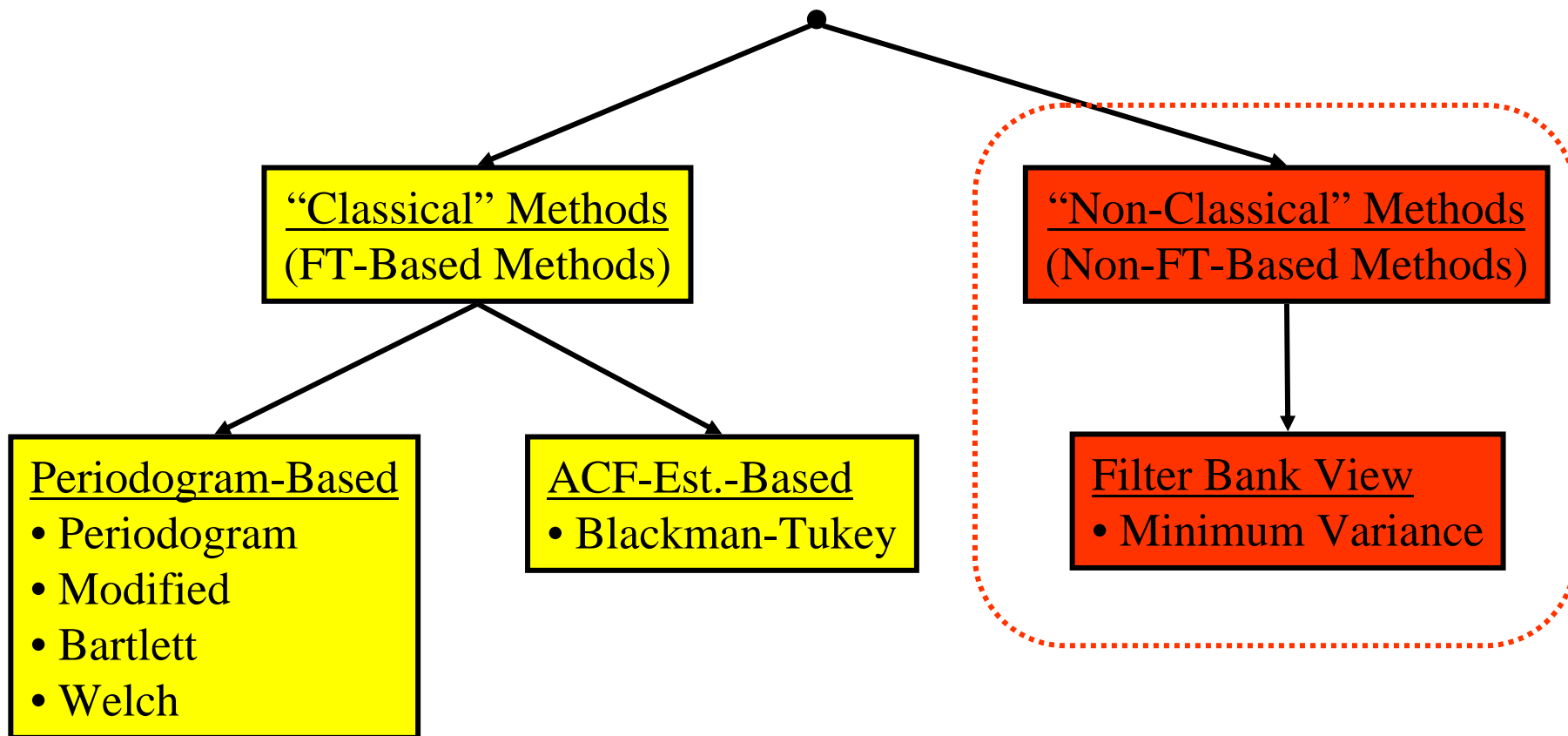


# Non-Classical Non-Parametric Methods

- Minimum Variance Method (MVSE)

LO-2.4.3, H-8.3

# Recall: Family of Non-Parametric Methods



# Minimum Variance Method

# Minimum Variance Method – Terminology

The Minimum Variance Spectral Estimation (MVSE) method has two other names:

- Maximum Likelihood Method (MLM)
- Capon's Method

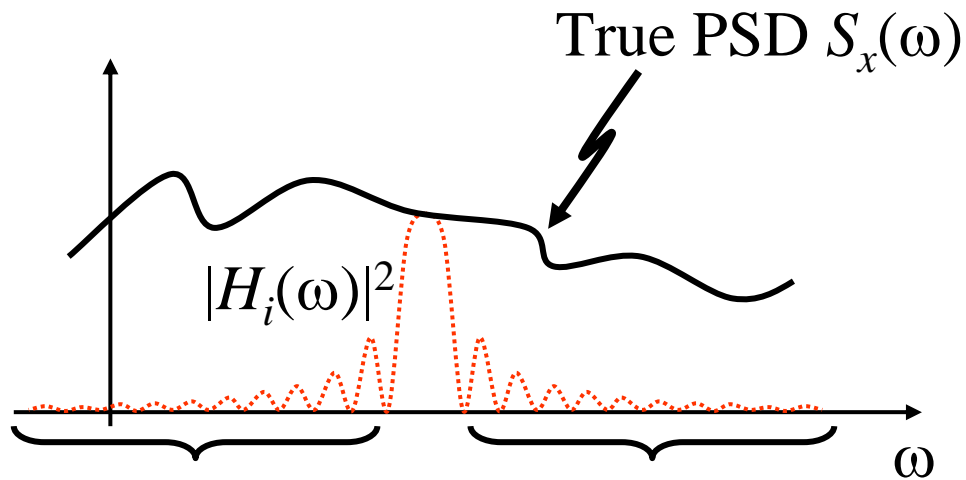
Note: The names MVSE and MLM are actually misnomers – this method:

- does NOT minimize the variance of the estimate
- does NOT maximize the “likelihood function”

# Recall: Filter Bank View of Periodogram

See Fig. 8.4 & 8.3 of Hayes

The problem is leakage from nearby frequencies:



Filter Sidelobes Leak  
“Out-of-Band” Power  
into Estimate at  $\omega_i$

# Goal for MVSE Method

Figure out a way to design each filter bank channel response to minimize the leakage – this is thus a data-dependent design.

Collect Data → “Design” Filters for Filter Bank

Want to “design” filters to minimize the sidelobes while keeping the mainlobe height at 1:

## “Design” Goals:

1. Want  $H_i(\omega_i) = 1$  ...to let through the desired  $S_x(\omega_i)$
2. Minimize total output power in the filter:

$$\rho_i = \frac{1}{2\pi} \int_{-\pi}^{\pi} |H_i(\omega)|^2 S_x(\omega) d\omega$$

This is equivalent to minimizing the sidelobe contribution even though the integral includes the desired  $\omega_i$

# Get Useable Form for $\rho$

The frequency response of filter  $h_i[n]$  is:  $H_i(\omega) = \sum_{n=0}^{p-1} h_i[n]e^{-j\omega n}$

Using this in the expression for  $\rho$  gives:

$$\rho_i = \frac{1}{2\pi} \int_{-\pi}^{\pi} \left[ \sum_{k=0}^{p-1} h_i[k]e^{-j\omega k} \right] \left[ \sum_{l=0}^{p-1} h_i^*[l]e^{j\omega l} \right] S_x(\omega) d\omega$$

$p = \text{Filter Length}$   
 $p < N$

$$= \sum_{k=0}^{p-1} \sum_{l=0}^{p-1} h_i[k]h_i^*[l] \underbrace{\frac{1}{2\pi} \int_{-\pi}^{\pi} S_x(\omega)e^{j\omega(l-k)} d\omega}_{=r_x[l-k]}$$

Now... Recognize as vector-matrix-vector multiplication

$$= \sum_{k=0}^{p-1} \sum_{l=0}^{p-1} h_i[k]h_i^*[l]r_x[l-k]$$

$\mathbf{R}_x = \text{Autocorrelation Matrix}$

$$= \mathbf{h}_i^H \mathbf{R}_x \mathbf{h}_i$$

“H” superscript = Hermitian Transpose  
= Transpose & Conjugate

# Autocorrelation Matrix

The AC matrix is the  $p \times p$  matrix whose  $i, j$  element is  $r_x[j - i]$ .

Example for  $p = 4$ :

$$\mathbf{R}_x = \begin{bmatrix} r_x[0] & r_x[1] & r_x[2] & r_x[3] \\ r_x[-1] & r_x[0] & r_x[1] & r_x[2] \\ r_x[-2] & r_x[-1] & r_x[0] & r_x[1] \\ r_x[-3] & r_x[-2] & r_x[-1] & r_x[0] \end{bmatrix}$$



## Now Minimize the Matrix Form:

For each  $i$ , minimize this:  $\rho_i = \mathbf{h}_i^H \mathbf{R}_x \mathbf{h}_i$

Under this constraint:

$$H_i(\omega_i) = 1 \Rightarrow \mathbf{h}_i^H \mathbf{e}_i = 1$$

$$\text{where } \mathbf{e}_i = [1 \quad e^{j\omega_i} \quad e^{j2\omega_i} \quad \dots \quad e^{j(p-1)\omega_i}]^T$$

Most common way to do constrained optimization is using the Lagrange Multiplier method:

$$J = \mathbf{h}_i^H \mathbf{R}_x \mathbf{h}_i - \lambda (\mathbf{h}_i^H \mathbf{e}_i - 1)$$

Lagrange says: Choose  $\mathbf{h}_i$  and  $\lambda$  to minimize  $J$

# Lagrange Minimization:

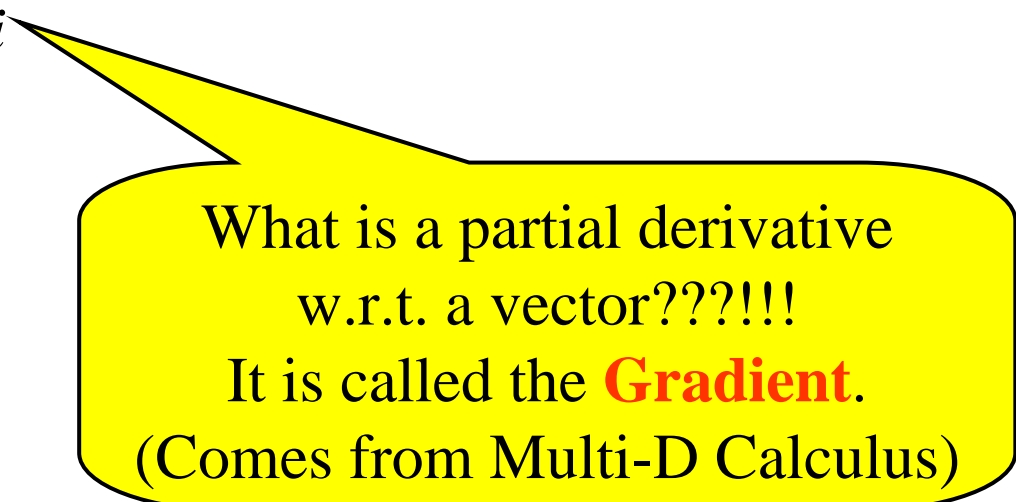
To find the  $\mathbf{h}_i$  and  $\lambda$  that minimizes  $J$ , *in general* set:

$$\frac{\partial J}{\partial \mathbf{h}_i} = \mathbf{0}^T \quad \& \quad \frac{\partial J}{\partial \lambda} = 0$$

But often an easier way is to do these two steps:

1. Do the partial w.r.t.  $\mathbf{h}_i$  and solve for  $\mathbf{h}_i$
2. Then choose  $\lambda$  to ensure solution meets the constraint

So... we need:  $\frac{\partial J}{\partial \mathbf{h}_i} = \mathbf{0}^T$



What is a partial derivative  
w.r.t. a vector????!!!  
It is called the **Gradient**.  
(Comes from Multi-D Calculus)

## Aside: Gradient

If  $g(\mathbf{x})$  is a scalar-valued function of a real-valued vector  $\mathbf{x}$ ,  
Then the gradient of  $g(\mathbf{x})$  is defined as:

$$\nabla_{\mathbf{x}}(g) = \frac{\partial g(\mathbf{x})}{\partial \mathbf{x}} = \left[ \frac{\partial g(\mathbf{x})}{\partial x_1} \quad \frac{\partial g(\mathbf{x})}{\partial x_2} \quad \dots \quad \frac{\partial g(\mathbf{x})}{\partial x_N} \right]$$

Gradient here is nothing more than: the vector whose elements are the partials w.r.t. each element of  $\mathbf{x}$ .

(Note: There are similar definitions when  $g(\mathbf{x})$  is vector-valued.)

“Example”  $g(\mathbf{x}) = \mathbf{c}^T \mathbf{x} = \sum_{i=1}^N c_i x_i = c_1 x_1 + c_2 x_2 + \dots + c_n x_N$

$$\begin{aligned} \frac{\partial g(\mathbf{x})}{\partial \mathbf{x}} &= \left[ \frac{\partial g(\mathbf{x})}{\partial x_1} \quad \frac{\partial g(\mathbf{x})}{\partial x_2} \quad \dots \quad \frac{\partial g(\mathbf{x})}{\partial x_N} \right] \\ &= [c_1 \quad c_2 \quad \dots \quad c_N] \\ &= \mathbf{c}^T \end{aligned}$$

# Lagrange Minimization (cont.):

Step #1: For our scalar-valued function  $J$  we get:

$$\underbrace{\frac{\partial J}{\partial \mathbf{h}_i}} = \mathbf{h}_{i,o}^H \mathbf{R}_x - \lambda \mathbf{e}_i^H \stackrel{\text{set}}{=} \mathbf{0}^T$$

Subscript “o”  
indicates  
“Optimal” – the  $\mathbf{h}$   
needed to get  $\mathbf{0}^T$

using standard results for gradients of common functions of vectors

Now solve this for  $\mathbf{h}_{i,o}$ :  $\mathbf{h}_{i,o}^H = \lambda \mathbf{e}_i^H \mathbf{R}_x^{-1}$

But...Depends on  $\lambda$ !!

Step #2: Choose  $\lambda$  to make this solution satisfy constraint:

$$\mathbf{h}_{i,o}^H \mathbf{e}_i = 1 \Rightarrow \left( \lambda \mathbf{e}_i^H \mathbf{R}_x^{-1} \right) \mathbf{e}_i = 1 \Rightarrow \lambda = \frac{1}{\mathbf{e}_i^H \mathbf{R}_x^{-1} \mathbf{e}_i}$$

.... Now use this  $\lambda$  in optimal  $\mathbf{h}_{i,o}$ :

$$\mathbf{h}_{i,o}^H = \frac{\mathbf{e}_i^H \mathbf{R}_x^{-1}}{\mathbf{e}_i^H \mathbf{R}_x^{-1} \mathbf{e}_i}$$



$$\mathbf{h}_{i,o} = \frac{\mathbf{R}_x^{-1} \mathbf{e}_i}{\mathbf{e}_i^H \mathbf{R}_x^{-1} \mathbf{e}_i}$$

# MVSE – Filter Solution

$$\mathbf{h}_{i,o}^H = \frac{\mathbf{e}_i^H \mathbf{R}_x^{-1}}{\mathbf{e}_i^H \mathbf{R}_x^{-1} \mathbf{e}_i} \longleftrightarrow \mathbf{h}_{i,o} = \frac{\mathbf{R}_x^{-1} \mathbf{e}_i}{\mathbf{e}_i^H \mathbf{R}_x^{-1} \mathbf{e}_i}$$

where  $\mathbf{e}_i = [1 \quad e^{j\omega_i} \quad e^{j2\omega_i} \quad \dots \quad e^{j(p-1)\omega_i}]$

This gives the optimal filter for estimating the power at the frequency  $\omega_i$

In *principle* – we need to solve this for each frequency  $\omega_i$  at which we wish to get a PSD estimate.

Then we would compute the output power at each filter and that would be our PSD estimate.

**BUT.... we have an equation for the output power:  $\rho_i$**

# MVSE – Power Estimate in Each Channel

The estimate of the power at  $\omega_i$  is nothing more than the minimized value of  $\rho_i$ :

$$\begin{aligned}\rho_{i,o} &= \mathbf{h}_{i,o}^H \mathbf{R}_x \mathbf{h}_{i,o} \\ &= \left[ \frac{\mathbf{e}_i^H \mathbf{R}_x^{-1}}{\mathbf{e}_i^H \mathbf{R}_x^{-1} \mathbf{e}_i} \right] \mathbf{R}_x \left[ \frac{\mathbf{e}_i^H \mathbf{R}_x^{-1}}{\mathbf{e}_i^H \mathbf{R}_x^{-1} \mathbf{e}_i} \right]^H = \left[ \frac{\mathbf{e}_i^H \mathbf{R}_x^{-1}}{\mathbf{e}_i^H \mathbf{R}_x^{-1} \mathbf{e}_i} \right] \mathbf{R}_x \left[ \frac{\mathbf{R}_x^{-1} \mathbf{e}_i}{\mathbf{e}_i^H \mathbf{R}_x^{-1} \mathbf{e}_i} \right] \\ &= \frac{\mathbf{e}_i^H \mathbf{R}_x^{-1} \mathbf{R}_x \mathbf{R}_x^{-1} \mathbf{e}_i}{(\mathbf{e}_i^H \mathbf{R}_x^{-1} \mathbf{e}_i)(\mathbf{e}_i^H \mathbf{R}_x^{-1} \mathbf{e}_i)} = \frac{\mathbf{e}_i^H \mathbf{R}_x^{-1} \mathbf{e}_i}{(\mathbf{e}_i^H \mathbf{R}_x^{-1} \mathbf{e}_i)(\mathbf{e}_i^H \mathbf{R}_x^{-1} \mathbf{e}_i)}\end{aligned}$$

Thus, the estimated power at frequency  $\omega_i$  is:

$$\hat{\sigma}_x^2(\omega_i) = \frac{1}{\mathbf{e}_i^H \mathbf{R}_x^{-1} \mathbf{e}_i}$$

# MVSE – PSD Estimate in Each Channel

To get the power spectral density we need to divide by the filter's bandwidth – for a filter of length  $p$  the BW is approximately  $1/p$  so our MVSE PSD estimate is:

$$\hat{S}_{MV}(\omega_i) = \frac{p}{\mathbf{e}_i^H \mathbf{R}_x^{-1} \mathbf{e}_i}$$

**But... this estimate requires the ACF in matrix form, which if we had it we'd probably know what the PSD is, too!!!**

**So... we need an estimate of the ACF in matrix form....**

# MVSE – Estimating The AC Matrix

$$\hat{\mathbf{R}}_x = \begin{bmatrix} \hat{r}_x[0] & \hat{r}_x[1] & \cdots & \hat{r}_x[p-1] \\ \hat{r}_x[-1] & \hat{r}_x[0] & \ddots & \\ \vdots & \ddots & \ddots & \hat{r}_x[1] \\ \hat{r}_x[-p+1] & \cdots & \hat{r}_x[-1] & \hat{r}_x[0] \end{bmatrix}$$

$$\hat{r}_x[k] = \frac{1}{N} \sum_{n=0}^{N-|k|-1} x[n+k]x^*[n]$$

Note: The  $p \times p$   
AC Matrix  
MUST  
be Estimated

$$\hat{S}_{MV}(\omega_i) = \frac{p}{\mathbf{e}_i^H \hat{\mathbf{R}}_x^{-1} \mathbf{e}_i}$$

Choose  $p < N$  so that high-order ACF lag estimates are reasonably accurate.

Note: There are other ways to estimate the AC Matrix!!!



# MVSE – Comments

## Implementation of MVSE:

Generally done directly on the data matrix  $\mathbf{X}$  for efficiency  
(see more advanced books)

Even with that, it is more complex than classical methods

## Performance of MVSE:

Provides better resolution than classical methods

Mostly used when spiky spectra are expected

(Although, the AR methods are usually better in that case)

If needed resolution can be met with classical – use them.

If not – consider either MVSE or Parametric Methods.