Least Squares Estimation, Filtering, and Prediction

- Principle of least squares
- Normal equations
- Weighted least squares
- Statistical properties
- FIR filters
- Windowing
- Combined forward-backward linear prediction
- Narrowband Interference Cancellation

Motivation

- If the second-order statistics are known, the optimum estimator is given by the normal equations
- In many applications, they aren't known
- Alternative approach is to estimate the coefficients from observed data
- Two possible approaches
  - Estimate required moments from available data and build an *approximate* MMSE estimator
  - Build an estimator that minimizes some error functional calculated from the available data

MMSE versus Least Squares

- Recall that MMSE estimators are optimal in expectation across the ensemble of all stochastic processes with the same second order statistics
- Least squares estimators minimize the error on a given **block** of data
  - In signal processing applications, the block of data is a finite-length period of time
- No guarantees about optimality on other data sets or other stochastic processes
- If the process is ergodic and stationary, the LSE estimator approaches the MMSE estimator as the size of the data set grows
  - This is the first time in this class we've discussed estimation from data
  - First time we need to consider ergodicity

Principle of Least Squares

- Will only discuss the sum of squares as the performance criterion
  - Recall our earlier discussion about alternatives
  - Essentially, picking sum of squares will permit us to obtain a closed-form optimal solution
- Requires a data set where both the inputs and desired responses are known
- Recall the range of possible applications
  - Plant modeling for control (system identification)
  - Inverse modeling/deconvolution
  - Interference cancellation
  - Prediction

Recalling the Book's Notation

- $y(n) \in \mathbb{C}^{1\times 1}$ is the **target** or **desired response**
- $x_k(n)$ represent the inputs
- These may be of several types
  - Multiple sensors, no lags: $\boldsymbol{x}(n) = [x_1(n), x_2(n), \dots, x_M(n)]^{\mathrm{T}}$
  - Lag window: $\boldsymbol{x}(n) = [x(n), x(n-1), \dots, x(n-M+1)]^{\mathrm{T}}$
  - Combined
- Data sets consists of values over the time span $0 \le n \le N-1$
- Boldface is now used for vectors and matrices
- The coefficients are represented as $\boldsymbol{c}(n)$

Change in Notation

In a trade of elegance and simplicity for inconsistency, I'm going to break with some of the book's notational conventions

$$\text{Notes: } \hat{y}(n) \triangleq \sum_{k=1}^{M} c_k(n) x_k(n) = \boldsymbol{c}^{\mathrm{T}}(n)\boldsymbol{x}(n)$$

$$\text{Book: } \hat{y}(n) \triangleq \sum_{k=1}^{M} c_k^*(n) x_k(n) = \boldsymbol{c}^{\mathrm{H}}(n)\boldsymbol{x}(n)$$

- In the case that $\boldsymbol{c}$ is real, they are consistent
- Rationale
  - The inner product notation leads to unnecessary complications in the notation
  - Most books use the same notation that the notes use
  - Leads to a symmetry: $\boldsymbol{c}^{\mathrm{T}}(n)\boldsymbol{x}(n) = \boldsymbol{x}^{\mathrm{T}}(n)\boldsymbol{c}(n)$

Definitions

$$\text{Estimate:} \quad \hat{y}(n) \triangleq \sum_{k=1}^{M} c_k(n) x_k(n) = \boldsymbol{c}^{\mathrm{T}}(n)\boldsymbol{x}(n)$$

$$\text{Estimation error:} \quad e(n) \triangleq y(n) - \hat{y}(n) \quad = y(n) - \boldsymbol{c}(n)\boldsymbol{x}(n)$$

$$\text{Sum of squared errors:} \quad E_e \triangleq \sum_{n=0}^{N-1} |e(n)|^2$$

- Coefficient vector $\boldsymbol{c}(n)$ is typically held constant over the data window, $0 \le n \le N-1$
- Contrast with adaptive filter approach
- The coefficients $\boldsymbol{c}$ that minimize E_e are called the **linear LSE estimator**

Matrix Formulation

$$\begin{bmatrix} e(0) \\ e(1) \\ \vdots \\ e(N\text{-}1) \end{bmatrix} = \begin{bmatrix} y(0) \\ y(1) \\ \vdots \\ y(N\text{-}1) \end{bmatrix} - \begin{bmatrix} x_1(0) & x_2(0) & \dots & x_M(0) \\ x_1(1) & x_2(1) & \vdots & x_M(1) \\ \vdots & \vdots & \ddots & \vdots \\ x_1(N\text{-}1) & x_2(N\text{-}1) & \dots & x_M(N\text{-}1) \end{bmatrix} \begin{bmatrix} c_1 \\ c_2 \\ \vdots \\ c_M \end{bmatrix}$$

$$\boldsymbol{e} = \boldsymbol{y} - \boldsymbol{X}\boldsymbol{c}$$

where

$$\boldsymbol{e} \triangleq \begin{bmatrix} e(0) & e(1) & \dots & e(N\text{-}1) \end{bmatrix}^{\mathrm{T}} \quad \text{error data vector } (N \times 1)$$

$$\boldsymbol{y} \triangleq \begin{bmatrix} y(0) & y(1) & \dots & y(N\text{-}1) \end{bmatrix}^{\mathrm{T}} \quad \text{desired response vector } (N \times 1)$$

$$\boldsymbol{X} \triangleq \begin{bmatrix} \boldsymbol{x}^{\mathrm{T}}(0) & \boldsymbol{x}^{\mathrm{T}}(1) & \dots & \boldsymbol{x}^{\mathrm{T}}(N\text{-}1) \end{bmatrix}^{\mathrm{T}} \quad \text{input data matrix } (N \times M)$$

$$\boldsymbol{c} \triangleq \begin{bmatrix} \boldsymbol{c}_1 & \boldsymbol{c}_2 & \dots & \boldsymbol{c}_M \end{bmatrix}^{\mathrm{T}} \quad \text{combiner parameter vector } (M \times 1)$$

Note: my definitions differ from the book by a conjugate factor *

Input Data Matrix Notation

$$\boldsymbol{X} = \begin{bmatrix} x_1(0) & x_2(0) & \dots & x_M(0) \\ x_1(1) & x_2(1) & \vdots & x_M(1) \\ \vdots & \vdots & \ddots & \vdots \\ x_1(N\text{-}1) & x_2(N\text{-}1) & \dots & x_M(N\text{-}1) \end{bmatrix} = \begin{bmatrix} \boldsymbol{x}^{\mathrm{T}}(0) \\ \boldsymbol{x}^{\mathrm{T}}(1) \\ \vdots \\ \boldsymbol{x}^{\mathrm{T}}(N-1) \end{bmatrix}$$
$$= [\vec{\boldsymbol{x}}_1 \quad \vec{\boldsymbol{x}}_2 \quad \dots \quad \vec{\boldsymbol{x}}_M]$$

- We will need to reference both the row and column vectors of the data matrix $\boldsymbol{X}$
- This is always awkward to denote
- Book's notation is redundant
  - Row vectors ("snapshots") indicated by (n) and boldface $\boldsymbol{x}$
  - Column vectors ("data records") indicated by $_k$ and $\vec{\boldsymbol{x}}$ (book uses $\tilde{\boldsymbol{x}}$)
- I don't know of a more elegant solution

Size of Data Matrix

$$\underset{N\times 1}{\boldsymbol{e}} = \underset{N\times 1}{\boldsymbol{y}} - \underset{N\times M}{\boldsymbol{X}}\underset{M\times 1}{\boldsymbol{c}}$$

- Suppose we wish to make $|\boldsymbol{e}| = 0$

$$\boldsymbol{y} = \boldsymbol{X}\boldsymbol{c}$$

- Suppose $\boldsymbol{X}$ has maximum rank
  - Linearly independent rows or columns
  - Always occurs with real data
- N linear equations and M unknowns
  - $N < M$: underdetermined, infinite number of solutions
  - $N = M$: unique solution, $\boldsymbol{c} = \boldsymbol{X}^{-1}\boldsymbol{y}$
  - $N > M$: overdetermined, no solution, in general
- In practical applications we pick $N > M$ (why?)

Block Processing

- LSE estimators *can* be used in **block processing mode**
  - Take a segment of N input-output observations, say $n_1 \leq n \leq n_1 + N - 1$
  - Estimate the coefficients
  - Increment the temporal location of the block to $n_1 + N_0$
- The blocks overlap by $N - N_0$ samples
- Reminiscent of Welch's method of PSD estimation
- Useful for parametric time-frequency analysis
- In most other nonstationary applications, adaptive filters are usually used instead

Normal Equations

$$\begin{aligned} E_e &= \|\boldsymbol{e}\|^2 = \boldsymbol{e}^{\mathrm{H}}\boldsymbol{e} \\ &= (\boldsymbol{y} - \boldsymbol{X}\boldsymbol{c})^{\mathrm{H}}(\boldsymbol{y} - \boldsymbol{X}\boldsymbol{c}) \\ &= (\boldsymbol{y}^{\mathrm{H}} - \boldsymbol{c}^{\mathrm{H}}\boldsymbol{X}^{\mathrm{H}})(\boldsymbol{y} - \boldsymbol{X}\boldsymbol{c}) \\ &= \boldsymbol{y}^{\mathrm{H}}\boldsymbol{y} - \boldsymbol{c}^{\mathrm{H}}\boldsymbol{X}^{\mathrm{H}}\boldsymbol{y} - \boldsymbol{y}^{\mathrm{H}}\boldsymbol{X}\boldsymbol{c} + \boldsymbol{c}^{\mathrm{H}}\boldsymbol{X}^{\mathrm{H}}\boldsymbol{X}\boldsymbol{c} \end{aligned}$$

- E_e is a nonlinear function of $\boldsymbol{y}$, $\boldsymbol{X}$, and $\boldsymbol{c}$
- Is a quadratic function of each of these components
- E_e is the sum of squared errors
  - Energy of the error signal over the interval $0 \leq n \leq N-1$
- If we take the average squared errors (ASE), we have an estimate of the mean square error = the estimated power of the error

$$\hat{P}_e = \frac{1}{N}E_e = \frac{1}{N}\sum_{n=0}^{N-1}|e(n)|^2$$

Normal Equation Components

Let us define

Average squared error (ASE): $\underset{1\times 1}{\hat{P}_e} \triangleq \frac{1}{N}\|\boldsymbol{e}\|^2 = \frac{1}{N}\sum_{n=0}^{N-1}|e(n)|^2$

Average signal power: $\underset{1\times 1}{\hat{P}_y} \triangleq \frac{1}{N}\|\boldsymbol{y}\|^2 = \frac{1}{N}\sum_{n=0}^{N-1}|y(n)|^2$

Average correlation: $\underset{M\times M}{\hat{\boldsymbol{R}}} \triangleq \frac{1}{N}\boldsymbol{X}^{\mathrm{H}}\boldsymbol{X} = \frac{1}{N}\sum_{n=0}^{N-1}\boldsymbol{x}(n)\boldsymbol{x}^{\mathrm{T}}(n)$

Average cross-correlation: $\underset{M\times 1}{\hat{\boldsymbol{d}}} \triangleq \frac{1}{N}\boldsymbol{X}^{\mathrm{H}}\boldsymbol{y} = \frac{1}{N}\sum_{n=0}^{N-1}\boldsymbol{x}(n)y(n)$

If we replaced the sample average $\frac{1}{N}\sum_{n=0}^{N-1}(\cdot)$ with expectation, $\mathrm{E}[\cdot]$, each of these terms would be the **mean ensemble value** rather than the **sample average estimate**

Relating LSE and MMSE Estimation

Then

$$\begin{aligned}\hat{P}_e &= \frac{1}{N}\left(\boldsymbol{y}^{\mathrm{H}}\boldsymbol{y} - \boldsymbol{c}^{\mathrm{H}}\boldsymbol{X}^{\mathrm{H}}\boldsymbol{y} - \boldsymbol{y}^{\mathrm{H}}\boldsymbol{X}\boldsymbol{c} + \boldsymbol{c}^{\mathrm{H}}\boldsymbol{X}^{\mathrm{H}}\boldsymbol{X}\boldsymbol{c}\right)\\ &= \hat{P}_y - \boldsymbol{c}^{\mathrm{H}}\hat{\boldsymbol{d}} - \hat{\boldsymbol{d}}^{\mathrm{H}}\boldsymbol{c} + \boldsymbol{c}^{\mathrm{H}}\hat{\boldsymbol{R}}\boldsymbol{c}\end{aligned}$$

This should look familiar...

If we assume $\hat{\boldsymbol{R}} > 0$, we can complete the square (fourth time!),

$$\begin{aligned}\hat{P}_e(\boldsymbol{c}) &= \hat{P}_y + \boldsymbol{c}^{\mathrm{H}}\hat{\boldsymbol{R}}(\boldsymbol{c} - \hat{\boldsymbol{R}}^{-1}\hat{\boldsymbol{d}}) - \hat{\boldsymbol{d}}^{\mathrm{H}}\boldsymbol{c} + \hat{\boldsymbol{d}}^{\mathrm{H}}\hat{\boldsymbol{R}}^{-1}\hat{\boldsymbol{d}} - \hat{\boldsymbol{d}}^{\mathrm{H}}\hat{\boldsymbol{R}}^{-1}\hat{\boldsymbol{d}}\\
&= \hat{P}_y + (\boldsymbol{c}^{\mathrm{H}})\hat{\boldsymbol{R}}(\boldsymbol{c} - \hat{\boldsymbol{R}}^{-1}\hat{\boldsymbol{d}}) - (\hat{\boldsymbol{d}}^{\mathrm{H}}\hat{\boldsymbol{R}}^{-1})\hat{\boldsymbol{R}}(\boldsymbol{c} - \hat{\boldsymbol{R}}^{-1}\hat{\boldsymbol{d}}) - \hat{\boldsymbol{d}}^{\mathrm{H}}\hat{\boldsymbol{R}}^{-1}\hat{\boldsymbol{d}}\\
&= \hat{P}_y + (\boldsymbol{c}^{\mathrm{H}} - \hat{\boldsymbol{d}}^{\mathrm{H}}\hat{\boldsymbol{R}}^{-1})\hat{\boldsymbol{R}}(\boldsymbol{c} - \hat{\boldsymbol{R}}^{-1}\hat{\boldsymbol{d}}) - \hat{\boldsymbol{d}}^{\mathrm{H}}\hat{\boldsymbol{R}}^{-1}\hat{\boldsymbol{d}}\\
&= \hat{P}_y - \hat{\boldsymbol{d}}^{\mathrm{H}}\hat{\boldsymbol{R}}^{-1}\hat{\boldsymbol{d}} + (\boldsymbol{c} - \hat{\boldsymbol{R}}^{-1}\hat{\boldsymbol{d}})^{\mathrm{H}}\hat{\boldsymbol{R}}(\boldsymbol{c} - \hat{\boldsymbol{R}}^{-1}\hat{\boldsymbol{d}})\\
&= \hat{P}_y - \hat{\boldsymbol{d}}^{\mathrm{H}}\hat{\boldsymbol{R}}^{-1}\hat{\boldsymbol{d}} + (\hat{\boldsymbol{R}}\boldsymbol{c} - \hat{\boldsymbol{d}})^{\mathrm{H}}\hat{\boldsymbol{R}}^{-1}(\hat{\boldsymbol{R}}\boldsymbol{c} - \hat{\boldsymbol{d}})\\
&= \hat{P}_{\mathrm{ls}} + (\hat{\boldsymbol{R}}\boldsymbol{c} - \hat{\boldsymbol{d}})^{\mathrm{H}}\hat{\boldsymbol{R}}^{-1}(\hat{\boldsymbol{R}}\boldsymbol{c} - \hat{\boldsymbol{d}})\end{aligned}$$

Least Squares Estimate

$$\hat{P}_e(\boldsymbol{c}) = \hat{P}_y - \hat{\boldsymbol{d}}^{\mathrm{H}}\hat{\boldsymbol{R}}^{-1}\hat{\boldsymbol{d}} + (\hat{\boldsymbol{R}}\boldsymbol{c} - \hat{\boldsymbol{d}})^{\mathrm{H}}\hat{\boldsymbol{R}}^{-1}(\hat{\boldsymbol{R}}\boldsymbol{c} - \hat{\boldsymbol{d}})$$

So the least squares estimate and minimum average squared error are given by

$$\begin{aligned}\boldsymbol{c}_{\mathrm{ls}} &= \hat{\boldsymbol{R}}^{-1}\hat{\boldsymbol{d}}\\ &= \left(\boldsymbol{X}^{\mathrm{H}}\boldsymbol{X}\right)^{-1}\boldsymbol{X}^{\mathrm{H}}\boldsymbol{y}\\ \hat{P}_{\mathrm{ls}} &\triangleq \hat{P}_y - \hat{\boldsymbol{d}}^{\mathrm{H}}\hat{\boldsymbol{R}}^{-1}\hat{\boldsymbol{d}}\\ &= \hat{P}_y - \hat{\boldsymbol{d}}^{\mathrm{H}}\boldsymbol{c}_{\mathrm{ls}}\end{aligned}$$

- Both the LSE and MSE criteria are quadratic functions of the coefficient vector $\boldsymbol{c}$

Computational Issues

$\mathcal{O}(M^2N)$	$\hat{\boldsymbol{R}} \triangleq \frac{1}{N}\boldsymbol{X}^{\mathrm{H}}\boldsymbol{X}$
$\mathcal{O}(MN)$	$\hat{\boldsymbol{d}} \triangleq \frac{1}{N}\boldsymbol{X}^{\mathrm{H}}\boldsymbol{y}$
$\mathcal{O}(M^3)$	$\boldsymbol{c}_{\mathrm{ls}} = \hat{\boldsymbol{R}}^{-1}\hat{\boldsymbol{d}}$

- Typically $M \ll N$
- The most computationally expensive operation is calculating the input correlation matrix!
- If $\boldsymbol{x}(n)$ comes from a stationary time series, we can speed this up with the FFT
  - Recall the estimate from ECE 538/638
  - More later

Geometric Derivation

$$\boldsymbol{e} = \boldsymbol{y} - \boldsymbol{X}\boldsymbol{c}$$

- I was disparaging of the geometric interpretation for the MSE case
- It makes a lot more sense for the LSE case
- We are trying to estimate an N dimensional vector as a linear combination of the M columns of X
- The orthogonal projection of $\boldsymbol{y}$ onto the M dimensional subspace minimizes the error

Inner Product Definition

- In this case an inner product can be defined as

$$\langle \vec{\boldsymbol{x}}_i, \vec{\boldsymbol{x}}_j \rangle \triangleq \vec{\boldsymbol{x}}_i^{\mathrm{H}} \vec{\boldsymbol{x}}_j = \sum_{n=0}^{N-1} x_i(n) x_j^*(n)$$

$$\|\vec{\boldsymbol{x}}_i\|^2 \triangleq \vec{\boldsymbol{x}}_i^{\mathrm{H}} \vec{\boldsymbol{x}}_i = \sum_{n=0}^{N-1} |x_i(n)|^2$$

- Can easily show that this has all of the required properties of an inner product

Projection Theorem: Estimator

- The projection theorem holds
  - Therefore the projection of $\boldsymbol{y}$ onto the column space of $\boldsymbol{X}$ minimizes the length of the residual vector

$$\|\boldsymbol{e}\|^2 = \sum_{n=0}^{N-1} |y(n) - \boldsymbol{c}^{\mathrm{T}} \boldsymbol{x}(n)|^2$$

  - The error vector is also orthogonal to the observations

$$\langle \boldsymbol{X}, \boldsymbol{y} - \boldsymbol{X}\boldsymbol{c}_{\mathrm{ls}} \rangle = \boldsymbol{X}^{\mathrm{H}} y - \boldsymbol{X}^{\mathrm{H}} \boldsymbol{X} \boldsymbol{c}_{\mathrm{ls}} = N\hat{\boldsymbol{d}} - N\hat{\boldsymbol{R}}\boldsymbol{c}_{\mathrm{ls}} = 0$$

- Thus we have another way of obtaining the normal equations

$$\hat{\boldsymbol{R}}\boldsymbol{c}_{\mathrm{ls}} = \hat{\boldsymbol{d}}$$

Projection Theorem: LSE

- Further, by orthogonality we have

$$\begin{aligned}
\|\boldsymbol{y}\|^2 &= \|\boldsymbol{e}\|^2 + \|\hat{\boldsymbol{y}}\|^2 \\
\|\boldsymbol{e}\|^2 &= \|\boldsymbol{y}\|^2 - \|\hat{\boldsymbol{y}}\|^2 \\
P_{\mathrm{ls}} &= P_y - \boldsymbol{c}_{\mathrm{ls}}^{\mathrm{H}} \boldsymbol{X}^{\mathrm{H}} \boldsymbol{X} \boldsymbol{c}_{\mathrm{ls}} \\
&= P_y - \hat{\boldsymbol{d}}^{\mathrm{H}} \hat{\boldsymbol{R}}^{-1} \hat{\boldsymbol{R}} \boldsymbol{c}_{\mathrm{ls}} \\
&= P_y - \hat{\boldsymbol{d}}^{\mathrm{H}} \boldsymbol{c}_{\mathrm{ls}}
\end{aligned}$$

Uniqueness

- As with the MMSE case we have
  - $\boldsymbol{c}_{\text{ls}}$ is unique if $\boldsymbol{X}$ has full column rank and $M \le N$
  - Otherwise there are infinite, equivalent solutions with the same LSE
- Regardless the LSE estimate $\hat{\boldsymbol{y}}(n) = \boldsymbol{c}^{\text{T}}(n)\boldsymbol{x}(n)$ is the same
- Full column rank is equivalent to the requirement that $\hat{\boldsymbol{R}}$ be positive definite

Weighted Least Squares

- Suppose some errors are more significant than others
- A closely related problem is weighted least squares

$$\begin{aligned} E_e &= \sum_{n=0}^{N-1} w_n^2 |y(n) - \boldsymbol{c}^{\text{T}}\boldsymbol{x}(n)|^2 \\ &= (\boldsymbol{y} - \boldsymbol{X}\boldsymbol{c})^{\text{H}}\boldsymbol{W}^2(\boldsymbol{y} - \boldsymbol{X}\boldsymbol{c}) \\ &= (\boldsymbol{W}\boldsymbol{y} - \boldsymbol{W}\boldsymbol{X}\boldsymbol{c})^{\text{H}}(\boldsymbol{W}\boldsymbol{y} - \boldsymbol{W}\boldsymbol{X}\boldsymbol{c}) \end{aligned}$$

If we define

$$\acute{\boldsymbol{y}} \triangleq \boldsymbol{W}\boldsymbol{y} \qquad\qquad \acute{\boldsymbol{X}} \triangleq \boldsymbol{W}\boldsymbol{X}$$

we have the original LSE problem

$$E_e = \sum_{n=0}^{N-1} |\acute{y}(n) - \boldsymbol{c}^{\text{T}}\acute{\boldsymbol{x}}(n)|^2 = (\acute{\boldsymbol{y}} - \acute{\boldsymbol{X}}\boldsymbol{c})^{\text{H}}(\acute{\boldsymbol{y}} - \acute{\boldsymbol{X}}\boldsymbol{c})$$

Weighted Least Squares Comments

$$E_e = (\boldsymbol{y} - \boldsymbol{X}\boldsymbol{c})^{\text{H}}\boldsymbol{W}^2(\boldsymbol{y} - \boldsymbol{X}\boldsymbol{c})$$

- Weighted least squares can be applied with any weighting matrix $\boldsymbol{W}^2$ that is positive definite
  - $\boldsymbol{W}^2$ does not have to be diagonal
  - All positive definite matrices have a square root $\boldsymbol{W}^2 = \boldsymbol{W}^{\text{H}}\boldsymbol{W}$ for some matrix $\boldsymbol{W}$
  - The square root is not unique
- Useful for
  - Windowing parametric estimators
  - Accounting for correlated observation noise

Statistical Models

$$\begin{aligned} \boldsymbol{y} &= \boldsymbol{X}\boldsymbol{c}_{\text{o}} + \boldsymbol{v} \\ \boldsymbol{c}_{\text{ls}} &= (\boldsymbol{X}^{\text{H}}\boldsymbol{X})^{-1}\boldsymbol{X}^{\text{H}}\boldsymbol{y} \\ &= (\boldsymbol{X}^{\text{H}}\boldsymbol{X})^{-1}\boldsymbol{X}^{\text{H}}\boldsymbol{X}\boldsymbol{c}_{\text{o}} + (\boldsymbol{X}^{\text{H}}\boldsymbol{X})^{-1}\boldsymbol{X}^{\text{H}}\boldsymbol{v} \\ &= \boldsymbol{c}_{\text{o}} + (\boldsymbol{X}^{\text{H}}\boldsymbol{X})^{-1}\boldsymbol{X}^{\text{H}}\boldsymbol{v} \end{aligned}$$

- Most of the statistical properties require a **statistical model** of how the data was generated
- This idea is similar to the state space model used by the Kalman filter
- The linear statistical model used here is more general (no assumption of state dynamics)
- Some of the properties don't hold when the model is not accurate

Linear Statistical Model

$$\underset{N\times 1}{\boldsymbol{y}} = \underset{N\times M}{\boldsymbol{X}}\underset{M\times 1}{\boldsymbol{c}_{\mathrm{o}}} + \underset{N\times 1}{\boldsymbol{v}}$$

- $\boldsymbol{y}$ is a vector of observed measurements
- $\boldsymbol{X}$ is well conditioned
  - Deterministic: $\boldsymbol{X}$ has full column rank
  - Stochastic: $\mathrm{E}[\boldsymbol{X}^{\mathrm{H}}\boldsymbol{X}]$ is positive definite
- $\boldsymbol{c}_{\mathrm{o}}$ is considered the "true" parameter vector
  - The notational overlap with MSE estimation notes is intentional
- $\boldsymbol{v}$ is a vector of random noise or "errors"
  - Note my notation differs from the text, which used $\boldsymbol{e}_o$
  - All of the elements of $\boldsymbol{X}$ are independent with $\boldsymbol{v}$
  - $\boldsymbol{v}$ has zero mean: $\mathrm{E}[\boldsymbol{v}] = 0$
  - The elements of $\boldsymbol{v}$ are uncorrelated: $\mathrm{E}[\boldsymbol{v}\boldsymbol{v}^{\mathrm{H}}] = \sigma_v^2\boldsymbol{I}$

Deterministic versus Stochastic Data Matrix

$$\boldsymbol{y} = \boldsymbol{X}\boldsymbol{c}_{\mathrm{o}} + \boldsymbol{v}$$

- The **data matrix** $\boldsymbol{X}$ may be deterministic or stochastic
- Appropriate model depends on the application
- Deterministic
  - Appropriate for conditions where the entries in $\boldsymbol{X}$ are controlled by the user
  - Only source of randomness is then $\boldsymbol{v}$
  - Simplifies the analysis of the estimator
- Stochastic
  - Appropriate for conditions where $\boldsymbol{X}$ are observed and randomly fluctuating
  - Most signal processing application

Deterministic Case: Estimator Properties

$$\boldsymbol{y} = \boldsymbol{X}\boldsymbol{c}_{\mathrm{o}} + \boldsymbol{v} \qquad \boldsymbol{c}_{\mathrm{ls}} = \boldsymbol{c}_{\mathrm{o}} + (\boldsymbol{X}^{\mathrm{H}}\boldsymbol{X})^{-1}\boldsymbol{X}^{\mathrm{H}}\boldsymbol{v}$$

- The estimator $\boldsymbol{c}_{\mathrm{ls}}$ is unbiased

$$\mathrm{E}[\boldsymbol{c}_{\mathrm{ls}}] = \boldsymbol{c}_{\mathrm{o}}$$

- The estimator covariance matrix is

$$\Lambda_{\mathrm{ls}} \triangleq \mathrm{E}\left[(\boldsymbol{c}_{ls} - \boldsymbol{c}_{\mathrm{o}})(\boldsymbol{c}_{ls} - \boldsymbol{c}_{\mathrm{o}})^{\mathrm{H}}\right] = \sigma_v^2(\boldsymbol{X}^{\mathrm{H}}\boldsymbol{X})^{-1} = \frac{\sigma_v^2}{N}\hat{\boldsymbol{R}}^{-1}$$

  - The diagonal elements of Λ_{ls} contain the variance of the elements of $\boldsymbol{c}_{\mathrm{ls}}$
  - If $\boldsymbol{v}$ is Gaussian, we can construct exact confidence intervals!
  - Difference in definition of $\hat{\boldsymbol{R}}$ make the dependence on N explicit
  - Covariance is inversely proportional to the number of observations

Deterministic Case: Residuals

$$\begin{aligned}
\boldsymbol{P} &\triangleq \boldsymbol{X}(\boldsymbol{X}^{\mathrm{H}}\boldsymbol{X})^{-1}\boldsymbol{X}^{\mathrm{H}} \\
\boldsymbol{c}_{\mathrm{ls}} &= \boldsymbol{c}_{\mathrm{o}} + (\boldsymbol{X}^{\mathrm{H}}\boldsymbol{X})^{-1}\boldsymbol{X}^{\mathrm{H}}\boldsymbol{v} \\
\boldsymbol{e} &= \boldsymbol{y} - \hat{\boldsymbol{y}} \\
&= \boldsymbol{y} - \boldsymbol{X}\boldsymbol{c}_{\mathrm{ls}} \\
&= \boldsymbol{y} - \boldsymbol{X}(\boldsymbol{X}^{\mathrm{H}}\boldsymbol{X})^{-1}\boldsymbol{X}^{\mathrm{H}}\boldsymbol{y} \\
&= \boldsymbol{X}\boldsymbol{c}_{\mathrm{o}} + \boldsymbol{v} - \boldsymbol{X}(\boldsymbol{X}^{\mathrm{H}}\boldsymbol{X})^{-1}\boldsymbol{X}(\boldsymbol{X}\boldsymbol{c}_{\mathrm{o}} + \boldsymbol{v}) \\
&= \boldsymbol{v} - \boldsymbol{P}\boldsymbol{v} \\
&= (\boldsymbol{I} - \boldsymbol{P})\boldsymbol{v}
\end{aligned}$$

Deterministic Case: Error Variance

$$
\begin{aligned}
\mathrm{E}_e &= \boldsymbol{e}^{\mathrm{H}}\boldsymbol{e} \\
&= \boldsymbol{v}^{\mathrm{H}}(\boldsymbol{I}-\boldsymbol{P})^{\mathrm{H}}(I-\boldsymbol{P})\boldsymbol{v} \\
&= \boldsymbol{v}^{\mathrm{H}}(\boldsymbol{I}-\boldsymbol{P})^{\mathrm{H}}(I-\boldsymbol{P})\boldsymbol{v} \\
&= \boldsymbol{v}^{\mathrm{H}}(\boldsymbol{I}-\boldsymbol{P})\boldsymbol{v} - \boldsymbol{v}^{\mathrm{H}}(\boldsymbol{P}-\boldsymbol{P}\boldsymbol{P})\boldsymbol{v} \\
&= \boldsymbol{v}^{\mathrm{H}}(\boldsymbol{I}-\boldsymbol{P})\boldsymbol{v} \\
&= \mathrm{trace}[\boldsymbol{v}^{\mathrm{H}}(\boldsymbol{I}-\boldsymbol{P})\boldsymbol{v}] \\
&= \mathrm{trace}[(\boldsymbol{I}-\boldsymbol{P})\boldsymbol{v}\boldsymbol{v}^{\mathrm{H}}] \\
\mathrm{E}[\mathrm{E}_e] &= \mathrm{trace}[(\boldsymbol{I}-\boldsymbol{P})\sigma_v^2\boldsymbol{I}] \\
&= \sigma_v^2\,\mathrm{trace}[\boldsymbol{I}-\boldsymbol{P}]
\end{aligned}
$$

Properties of the $\mathrm{trace}[\cdot]$ operator

$$\mathrm{trace}[\boldsymbol{A}\boldsymbol{B}] = \mathrm{trace}[\boldsymbol{B}\boldsymbol{A}] \qquad \mathrm{trace}[\boldsymbol{A}+\boldsymbol{B}] = \mathrm{trace}[\boldsymbol{A}] + \mathrm{trace}[\boldsymbol{B}]$$

Deterministic Case: Error Variance

$$
\begin{aligned}
\mathrm{E}[\mathrm{E}_e] &= \sigma_v^2\,\mathrm{trace}[I-\boldsymbol{P}] \\
\mathrm{trace}[I-\boldsymbol{P}] &= \mathrm{trace}[I-\boldsymbol{X}(\boldsymbol{X}^{\mathrm{H}}\boldsymbol{X})^{-1}\boldsymbol{X}^{\mathrm{H}}] \\
&= \mathrm{trace}[I] - \mathrm{trace}[(\boldsymbol{X}^{\mathrm{H}}\boldsymbol{X})^{-1}\boldsymbol{X}^{\mathrm{H}}\boldsymbol{X}] \\
&= N - \mathrm{trace}[I] \\
&= N - M \\
\mathrm{E}[\mathrm{E}_e] &= \sigma_v^2(N-M) \\
\hat{\sigma}_v^2 &\triangleq \frac{1}{N-M}\,\mathrm{E}_e \\
\mathrm{E}[\hat{\sigma}_v^2] &= \frac{1}{N-M}\,\mathrm{E}[\mathrm{E}_e] \\
&= \mathrm{E}_e
\end{aligned}
$$

- Therefore $\hat{\sigma}_v^2$ is unbiased
- The difference $N-M$ is often called the **degrees of freedom**

Deterministic Case: Other, Unproved Properties

$$\boldsymbol{y} = \boldsymbol{X}\boldsymbol{c}_{\mathrm{o}} + \boldsymbol{v} \qquad \boldsymbol{c}_{\mathrm{ls}} = \boldsymbol{c}_{\mathrm{o}} + (\boldsymbol{X}^{\mathrm{H}}\boldsymbol{X})^{-1}\boldsymbol{X}^{\mathrm{H}}\boldsymbol{v}$$

- The weighted LSE estimate is also unbiased
- If the error covariance is not diagonal, $R_v \neq \sigma_v^2 I$, the optimal estimator is obtained by setting $\boldsymbol{W}^2 = \boldsymbol{R}_v^{-1}$
- In both cases, the LSE estimator is the **best linear unbiased estimator** (BLUE)
  - Of all the unbiased estimators, this one has the smallest variance
- If $\boldsymbol{v}$ is normally distributed,, the LSE estimator is also the maximum likelihood estimator

Stochastic Case: Properties

$$\boldsymbol{y} = \boldsymbol{X}\boldsymbol{c}_{\mathrm{o}} + \boldsymbol{v} \qquad \boldsymbol{c}_{\mathrm{ls}} = \boldsymbol{c}_{\mathrm{o}} + (\boldsymbol{X}^{\mathrm{H}}\boldsymbol{X})^{-1}\boldsymbol{X}^{\mathrm{H}}\boldsymbol{v}$$

- Model assumptions
  - $\boldsymbol{X}$ and $\boldsymbol{v}$ are statistically independent
  - Merely uncorrelated is insufficient in this case
- The estimator is still unbiased in this case
- The covariance of $\boldsymbol{c}_{\mathrm{ls}}$ is given by

$$\boldsymbol{\Lambda}_{\mathrm{ls}} \triangleq \mathrm{E}\left[(\boldsymbol{c}_{\mathrm{ls}}-\boldsymbol{c}_{\mathrm{o}})(\boldsymbol{c}_{\mathrm{ls}}-\boldsymbol{c}_{\mathrm{o}})^{\mathrm{H}}\right] = \sigma_v^2\,\mathrm{E}[(\boldsymbol{X}^{\mathrm{H}}\boldsymbol{X})^{-1}]$$

- Proofs are in the text

Least Squares Filters

$$
\begin{aligned}
e(n) &= y(n) - \sum_{k=0}^{M-1} h(k)x(n-k) \\
&= y(n) - \boldsymbol{c}^{\mathrm{T}}\boldsymbol{x}(n)
\end{aligned}
$$

$$
\boldsymbol{x}(n) \triangleq \begin{bmatrix} \boldsymbol{x}(n) & \boldsymbol{x}(n-1) & \dots & \boldsymbol{x}(n-M+1) \end{bmatrix}^{\mathrm{T}}
$$

- Several things change we we consider the lagged window case
  - The observations are stochastic, not deterministic
  - The estimated correlation matrix $\hat{\boldsymbol{R}}$ has a relationship to the estimated correlation of the stochastic process
  - Edge effects of the signals mean our input vectors are not always complete

Edge Conditions

$$
\underbrace{\begin{bmatrix} e(0) \\ e(1) \\ \vdots \\ e(M\text{-}2) \\ \hline e(M\text{-}1) \\ \vdots \\ e(N\text{-}1) \\ \hline e(N) \\ \vdots \\ e(N{+}M\text{-}3) \\ e(N{+}M\text{-}2) \end{bmatrix}}_{(N+M-1)\times 1}
=
\underbrace{\begin{bmatrix} y(0) \\ y(1) \\ \vdots \\ y(M\text{-}2) \\ \hline y(M\text{-}1) \\ \vdots \\ y(N\text{-}1) \\ \hline 0 \\ \vdots \\ 0 \\ 0 \end{bmatrix}}_{(N+M-1)\times 1}
-
\underbrace{\begin{bmatrix} x(0) & 0 & \dots & 0 \\ x(1) & x(0) & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ x(M\text{-}2) & x(M\text{-}3) & \dots & 0 \\ \hline x(M\text{-}1) & x(M\text{-}2) & \dots & x(0) \\ \vdots & \vdots & \ddots & \vdots \\ x(N\text{-}1) & x(N\text{-}2) & \dots & x(N\text{-}M) \\ \hline 0 & x(N\text{-}1) & \dots & x(N\text{-}M{+}1) \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & x(N\text{-}2) \\ 0 & 0 & \dots & x(N\text{-}1) \end{bmatrix}}_{(N+M-1)\times M}
\underbrace{\begin{bmatrix} c_0 \\ c_1 \\ \vdots \\ c_{M\text{-}1} \end{bmatrix}}_{M\times 1}
$$

Computing the Correlation Matrix

$$
\hat{r}_{i+1,j+1} = x^*(N_i - i)x(N_i - j) - x^*(N_f + 1 - i)x(N_f + 1 - j) + \hat{r}_{ij}
$$

- The data matrix $\boldsymbol{X}$ is toeplitz
- This gives a structure to $\boldsymbol{X}^{\mathrm{H}}\boldsymbol{X}$ that can be used to increase computational efficiency
- However, $\hat{R}$ is not necessarily toeplitz
  - Depends on the data matrix

Derivation of Correlation Matrix Recursion

$$
\begin{aligned}
\hat{r}_{ij} &= \sum_{n=N_i}^{N_f} x^*(n+1-i)x(n+1-j) \\
\hat{r}_{i+1,j+1} &= \sum_{n=N_i}^{N_f} x^*\left(n+1-(i+1)\right)x\left(n+1-(j+1)\right) \\
&= \sum_{n=N_i-1}^{N_f-1} x^*(n+1-i)x(n+1-j) \\
&= x^*(N_i-1+1-i)x(N_i-1+1-j) \\
&\quad -x^*(N_f+1-i)x(N_f+1-j) + \sum_{n=N_i}^{N_f} x^*(n+1-i)x(n+1-j) \\
&= x^*(N_i-i)x(N_i-j) - x^*(N_f+1-i)x(N_f+1-j) + \hat{r}_{ij}
\end{aligned}
$$

Correlation Matrix Recursions

$$\hat{r}_{i+1,j+1} = x^*(N_i - i)x(N_i - j) - x^*(N_f + 1 - i)x(N_f + 1 - j) + \hat{r}_{ij}$$

- Once the first row of $\hat{R}$ is calculated, the recursion above can be used to fill out the rest of the matrix
- Reduces the computation from $\mathcal{O}(NM^2)$ to $\mathcal{O}(NM)$
- Can reduce even further to $\mathcal{O}(N \log N)$ via the FFT if the first row is equivalent to the signal correlation estimates discussed last term

Windowing

$$E_e = \sum_{n=N_i}^{N_f} |e(n)|^2 = \boldsymbol{e}^{\mathrm{H}}\boldsymbol{e}$$

There are four ways to select the range for LSE estimation

- **Prewindowing**: $N_i = 0$ and $N_f = N - 1$
  - Essentially treats $x(-1), \dots, x(-M+1)$ equal to zero
  - Used in adaptive filters mostly for sake of simplicity
- **Postwindowing**: $N_i = M - 1$ and $N_f = N + M - 2$
  - No one uses this
  - Included for completeness only

Windowing Continued

- **Short/No Windowing**: $N_i = M - 1$ and $N_f = N - 1$
  - Use only available data
  - No artificial data
  - No distortions
  - Unbiased, highest variance
  - Sometimes called the **autocorrelation method**
- **Tall/Full windowing**: $N_i = 0$ and $N_f = N + M - 2$
  - $\hat{\boldsymbol{R}}$ becomes toeplitz (efficient order recursions)
  - Sometimes called the **covariance method**
  - Equivalent to calculating $\hat{\boldsymbol{R}}$ with the biased signal correlation estimate (ECE 5/638)
  - Tip: make sure data is zero mean or detrended

Unbiased Autocorrelation Estimate?

- Full windowing is equivalent to using the biased correlation estimate discussed last term
- Conceptually could also use the unbiased estimate
- Not discussed in text
- Properties
  - $\hat{\boldsymbol{R}}$ is toeplitz (by construction)
  - $\hat{\boldsymbol{R}}$ may not be positive definite
  - Estimated AR process models may be unstable — but does it matter?
  - Uses all of the data to calculate every element of $\hat{\boldsymbol{R}}$

Forward and Backward Linear Prediction (FBLP)

$$\hat{x}_{\mathrm{f}}(n) = -\sum_{k=1}^{M} a_k(n)x(n-k) = -\boldsymbol{a}^{\mathrm{T}}(n)\boldsymbol{x}(n-1)$$

$$\hat{x}_{\mathrm{b}}(n) = -\sum_{k=0}^{M-1} b_k(n)x(n-k) = -\boldsymbol{b}^{\mathrm{T}}(n)\boldsymbol{x}(n)$$

$$e_{\mathrm{f}} = x(n) + \sum_{k=1}^{M} a_k(n)x(n-k) = x(n) + \boldsymbol{a}^{\mathrm{T}}(n)\boldsymbol{x}(n-1)$$

$$e_{\mathrm{b}} = \sum_{k=0}^{M-1} b_k(n)x(n-k) + x(n-M) = \boldsymbol{b}^{\mathrm{T}}(n)\boldsymbol{x}(n) + x(n-M)$$

- Recall that for *stationary* stochastic processes, the optimum MMSE forward and backward linear predictors have conjugate symmetry

$$\boldsymbol{a}_{\mathrm{o}} = \boldsymbol{J}\boldsymbol{b}_{\mathrm{o}}^*$$

Forward and Backward Linear Prediction Continued

$$\boldsymbol{a}_{\mathrm{o}} = \boldsymbol{J}\boldsymbol{b}_{\mathrm{o}}^*$$

- This symmetry stems from the Toeplitz structure of the autocorrelation matrix
- If the estimated autocorrelation matrix doesn't have it, perhaps we could improve performance by minimize the forward and backward sum of squared errors

Forward and Backward Linear Prediction Continued

$$\underset{(N+M)\times(M+1)}{\bar{\boldsymbol{X}}} \triangleq \begin{bmatrix} x(0) & 0 & \dots & 0 \\ x(1) & x(0) & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ x(M) & x(M-1) & \dots & x(0) \\ \vdots & \vdots & \ddots & \vdots \\ x(N-1) & x(N-2) & \dots & x(N-M-1) \\ 0 & x(N-1) & \dots & x(N-M) \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & x(N-1) \end{bmatrix}$$

$$\boldsymbol{e}_{\mathrm{f}} \triangleq \bar{\boldsymbol{X}} \begin{bmatrix} 1 \\ \boldsymbol{a}_{\mathrm{fb}} \end{bmatrix} \qquad \boldsymbol{e}_{\mathrm{b}}^* \triangleq \bar{\boldsymbol{X}}^* \begin{bmatrix} \boldsymbol{b}^* \\ 1 \end{bmatrix} = \bar{\boldsymbol{X}}^*\boldsymbol{J} \begin{bmatrix} 1 \\ \boldsymbol{a}_{\mathrm{fb}} \end{bmatrix}$$

Forward and Backward Linear Prediction Continued

$$\begin{aligned} \begin{bmatrix} \boldsymbol{e}_{\mathrm{f}} \\ \boldsymbol{e}_{\mathrm{b}}^* \end{bmatrix} &= \begin{bmatrix} \bar{\boldsymbol{X}} \\ \bar{\boldsymbol{X}}^*\boldsymbol{J} \end{bmatrix} \begin{bmatrix} 1 \\ \boldsymbol{a}_{\mathrm{fb}} \end{bmatrix} \\ &= \boldsymbol{x}_{\mathrm{fb}} + \begin{bmatrix} \boldsymbol{X} \\ \boldsymbol{X}^*\boldsymbol{J} \end{bmatrix} \left[\boldsymbol{a}_{\mathrm{fb}}\right] \end{aligned}$$

$$
\begin{bmatrix} e_\mathrm{f} \\ e_\mathrm{b}^* \end{bmatrix} =
\left[\begin{array}{c}
x(0) \\ \vdots \\ x(M\text{-}1) \\ \hline
x(M) \\ \vdots \\ x(N\text{-}1) \\ \hline
0 \\ \vdots \\ 0 \\ \hline
0 \\ \vdots \\ 0 \\ \hline
x^*(0) \\ \vdots \\ x^*(N\text{-}M\text{-}1) \\ \hline
x^*(N\text{-}M) \\ \vdots \\ x^*(N\text{-}1)
\end{array}\right]
+
\left[\begin{array}{ccccc}
0 & 0 & \dots & 0 & 0 \\
\vdots & \vdots & \ddots & \vdots & \vdots \\
x(M\text{-}2) & x(M\text{-}3) & \dots & x(0) & 0 \\ \hline
x(M\text{-}1) & x(M\text{-}2) & \dots & x(1) & x(0) \\
\vdots & \vdots & \ddots & \vdots & \vdots \\
x(N\text{-}2) & x(N\text{-}3) & \dots & x(N\text{-}M) & x(N\text{-}M\text{-}1) \\ \hline
x(N\text{-}1) & x(N\text{-}2) & \dots & x(N\text{-}M\text{+}1) & x(N\text{-}M) \\
\vdots & \vdots & \ddots & \vdots & \vdots \\
0 & 0 & \dots & 0 & x(N\text{-}1) \\ \hline
0 & 0 & \dots & 0 & x^*(0) \\
\vdots & \vdots & \ddots & \vdots & \vdots \\
x^*(0) & x^*(1) & \dots & x^*(M\text{-}2) & x^*(M\text{-}1) \\ \hline
x^*(1) & x^*(2) & \dots & x^*(M\text{-}1) & x^*(M) \\
\vdots & \vdots & \ddots & \vdots & \vdots \\
x^*(N\text{-}M) & x^*(N\text{-}M\text{+}1) & \dots & x^*(N\text{-}2) & x^*(N\text{-}1) \\ \hline
x^*(N\text{-}M\text{+}1) & x^*(N\text{-}M\text{+}2) & \dots & x^*(N\text{-}1) & 0 \\
\vdots & \vdots & \ddots & \vdots & \vdots \\
0 & 0 & \dots & 0 & 0
\end{array}\right]
\boldsymbol{a}_\mathrm{fb}
$$

Forward and Backward Linear Prediction Continued

$$
\boldsymbol{e}_\mathrm{fb} = \boldsymbol{x}_\mathrm{fb} + \boldsymbol{X}_\mathrm{fb}\boldsymbol{a}_\mathrm{fb} \qquad \boldsymbol{x}_\mathrm{fb} \triangleq \begin{bmatrix} \boldsymbol{x} \\ \boldsymbol{0} \\ \boldsymbol{0} \\ \boldsymbol{x}^* \end{bmatrix} \qquad \boldsymbol{X}_\mathrm{fb} \triangleq \begin{bmatrix} \boldsymbol{0} \\ \boldsymbol{X} \\ \boldsymbol{X}^*\boldsymbol{J} \\ \boldsymbol{0} \end{bmatrix}
$$

We've already solved this problem

$$
\boldsymbol{a}_\mathrm{ls} = -\left(\boldsymbol{X}_\mathrm{fb}^\mathrm{H}\boldsymbol{X}_\mathrm{fb}\right)^{-1}\boldsymbol{X}_\mathrm{fb}^\mathrm{H}\boldsymbol{x}_\mathrm{fb}
$$

$$
\begin{aligned}
\boldsymbol{X}_\mathrm{fb}^\mathrm{H}\boldsymbol{X}_\mathrm{fb} &= \begin{bmatrix} \boldsymbol{0} & \boldsymbol{X}^\mathrm{H} & \boldsymbol{J}\boldsymbol{X}^\mathrm{T} & \boldsymbol{0} \end{bmatrix} \begin{bmatrix} \boldsymbol{0} \\ \boldsymbol{X} \\ \boldsymbol{X}^*\boldsymbol{J} \\ \boldsymbol{0} \end{bmatrix} \\
&= \boldsymbol{X}^\mathrm{H}\boldsymbol{X} + \boldsymbol{J}\boldsymbol{X}^\mathrm{T}\boldsymbol{X}^*\boldsymbol{J} \\
&\propto \hat{\boldsymbol{R}} + \boldsymbol{J}\hat{\boldsymbol{R}}^*\boldsymbol{J} \\
\boldsymbol{X}_\mathrm{fb}^\mathrm{H}\boldsymbol{x}_\mathrm{fb} &\propto \hat{\boldsymbol{r}}_\mathrm{f} + \hat{\boldsymbol{r}}_\mathrm{f}^*
\end{aligned}
$$

Forward and Backward Linear Prediction Continued

$$
\hat{\boldsymbol{R}}_\mathrm{fb} = \boldsymbol{X}^\mathrm{H}\boldsymbol{X} + \boldsymbol{J}\boldsymbol{X}^\mathrm{T}\boldsymbol{X}^*\boldsymbol{J}
$$

- Makes the autocorrelation matrix more symmetric

$$
\hat{\boldsymbol{R}}_\mathrm{fb} = \boldsymbol{J}\hat{\boldsymbol{R}}_\mathrm{fb}\boldsymbol{J}
$$

- If no windowing is used, is sometimes called the **modified covariance method**
- With full windowing

$$
\boldsymbol{a}_\mathrm{fb} = (\boldsymbol{a} + \boldsymbol{J}\boldsymbol{b}^*)/2
$$

- Works really well for AR signal modeling and parametric spectral estimation (more later)

Summary of Least Squares Filter Estimation

Technique	PD	Unbiased	FFT	Toeplitz	WLS
Pre-Windowing	✓				✓
Post-Windowing	✓				✓
Full-Windowing	✓		✓	✓	✓
Short-Windowing	✓	✓			✓
Unbiased		✓	✓		
Forward/Backward	✓	$\ast$		$\star$	✓

$\ast$:Can be, depending on windowing technique used. $\star$:Persymmetric, $\hat{\boldsymbol{R}} = \boldsymbol{J}\hat{\boldsymbol{R}}\boldsymbol{J}$.

- Forward/backward limited to one-step prediction applications and stationary segments.
  - Works best when these conditions are met
- Otherwise short-windowing or unbiased is probably best, depending on importance of $\hat{\boldsymbol{R}}$ being positive definite

Narrowband Inteference

- Many types of signals are corrupted by **narrowband interference**
- Typically electrical noise from electrical lines, radio frequency interference, and electrical equipment
- Sensors can pick up this noise through many different means
  - Closed loops (magnetic coupling)
  - Capacitive (electrostatic) coupling
  - Electromagnetic radiation (wires = antennas)
  - Power line interference
  - Voltage drops on ground lines or power lines

Narrowband Inteference Properties

$$x(n) = s(n) + y(n) + v(n)$$

where

$$\begin{aligned} s(n) &= \text{ signal of interest} \\ y(n) &= \text{ narrowband interference} \\ v(n) &= \text{ white noise} \end{aligned}$$

- In many applications signal is broadband and unpredictable
- Cannot be separated from white noise with a linear filter (spectral overlap)
- Narrowband interfence is predictable
- Consists of sharp peaks in the frequency domain
- Can be eliminated with notch filters, but must know the frequencies

Narrowband Inteference Cancellation Concept

$$x(n) = s(n) + y(n) + v(n)$$

Suppose all three components of $x(n)$ are mutually uncorrelated

$$\begin{aligned} r_x(\ell) &\triangleq \mathrm{E}[x(n)x^*(n-\ell)] \\ &= \mathrm{E}[(s(n) + y(n) + v(n))\,(s^*(n-\ell) + y^*(n-\ell) + v^*(n-\ell))] \\ &= r_s(\ell) + r_y(\ell) + \sigma_v^2\delta(\ell) \end{aligned}$$

- Suppose $r_s(\ell) \approx 0$ for $\ell > D$ and $r_y(\ell) \neq 0$ for $\ell > D$
- This implies $s(n)$ is broadband and $r_y(\ell)$ is "narrowband"
- This means if we try to predict $x(n)$ D steps ahead, only part of the $y(n)$ component will be (partially) predictable

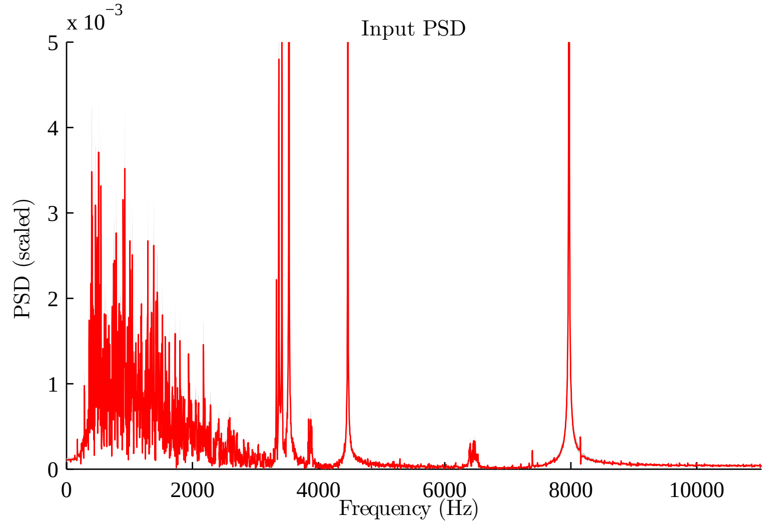
$$\hat{x}_{n-D}(n) = \hat{y}_{n-D}(n)$$

- The difference will then contain less of the interference

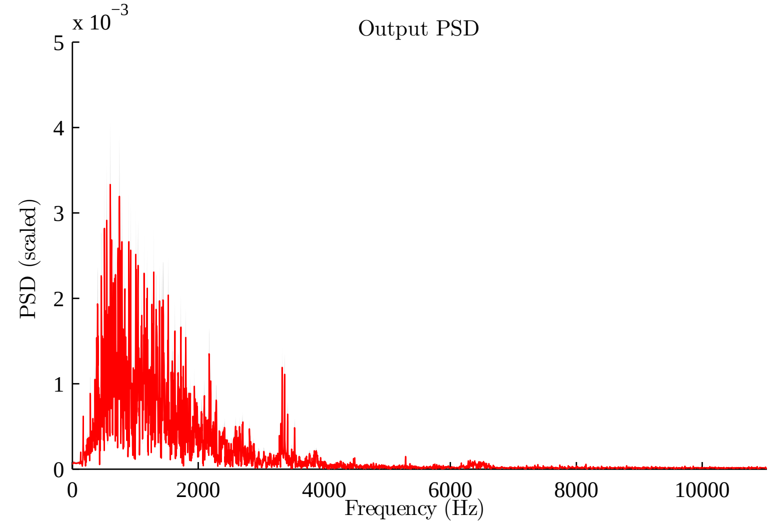
Example 1: Microelectrode Narrowband Interference

- Microelectrode recordings often contain narrowband interference
- The signal of interest are spikes that can be modeled as a point process
- In most cases, the spikes are not predictable with linear filters
- The duration of the spikes is typically 1 ms
- Use a narrowband interference canceller to eliminate the narrowband interference

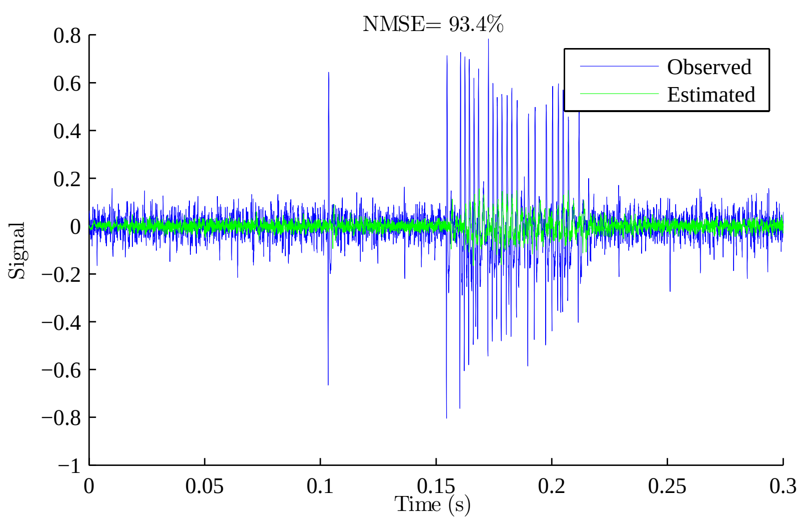
Example 1: Signal PSD



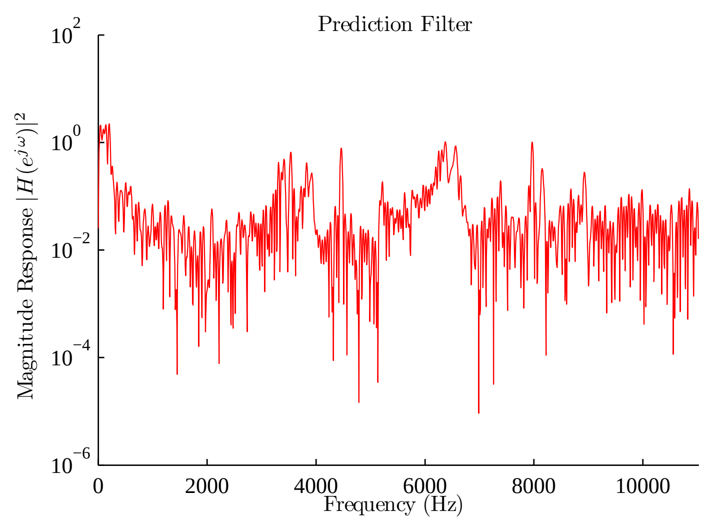
Example 1: Output PSD



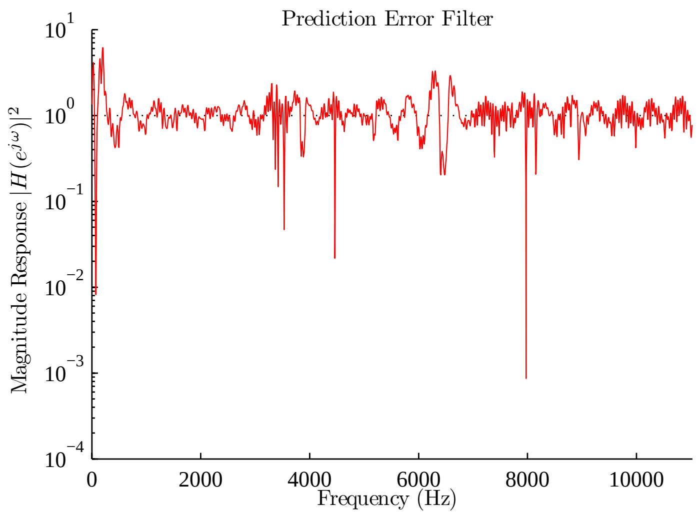
Example 1: Input and Predicted Signal



Example 1: Prediction Filter Frequency Response



Example 1: Prediction Error Filter Frequency Response



Example 1: MATLAB Code

```
clear all;
close all;

%==================================================================
% Parameters
%==================================================================
T  = 5;                                % Signal duration (s)
td = 2e-3;                             % Prediction delay (s)
tf = 50e-3;                            % Filter window duration (s)

%==================================================================
% Load the Data
%==================================================================
load MER.mat;

%==================================================================
% Preprocessing
%==================================================================
nx = ceil(fs*T);                       % Duration of extracted segment
d  = round(td*fs);                     % Delay in units of samples
fo = round(tf*fs);
y = x(d+(1:nx));                       % Extract the target output
x = x(1:nx);                           % Extract a segment of the signal

my = mean(y);                          % Mean of target signal

%==================================================================
% Estimate the Coefficients
%==================================================================
[c,yh] = LeastSquaresFilter(x,y,fs,500,1,1);

%==================================================================
% Post Processing
%==================================================================
```



```
hpe = [1;zeros(d-1,1);-c];             % Impulse response of the prediction error filter
e   = y-yh;                            % Residuals

%==================================================================
% Figures
%==================================================================
BlackmanTukey(y,fs,1);
FigureSet(1,'Slides');
ylim([0 0.005]);
title('Input PSD');
AxisSet(8);
print('NBISignalPSD','-depsc');

BlackmanTukey(y-yh,fs,1);
FigureSet(1,'Slides');
ylim([0 0.005]);
title('Output PSD');
AxisSet(8);
print('NBIOutputPSD','-depsc');

figure;
FigureSet(1,'Slides');
k = 1:nx;
t = (k-0.5)/fs;
h = plot(t,y,'b',t,yh,'g');
set(h,'LineWidth',0.2);
xlim([0 0.3]);
xlabel('Time (s)');
ylabel('Signal');
legend('Observed','Estimated');
set(get(gca,'Title'),'Interpreter','LaTeX');
title(sprintf('NMSE=%5.1f\\%%',100*sum((y-yh).^2)/sum((y-my).^2)));
box off;
AxisSet(8);
print('NBISignalEstimate','-depsc');

figure;
```



```
FigureSet(2,'Slides');
[h,f] = freqz(c,1,2^11,fs);
h = semilogy(f,abs(h).^2,'r');
set(h,'LineWidth',0.4);
xlim([0 fs/2]);
xlabel('Frequency (Hz)');
set(get(gca,'YLabel'),'Interpreter','LaTeX');
ylabel('Magnitude Response $|H(e^{j\omega})|^2$');
title('Prediction Filter');
box off;
AxisSet(8);
print('NBIPredictionFrequencyResponse','-depsc');

figure;
FigureSet(2,'Slides');
[h,f] = freqz(hpe,1,2^11,fs);
semilogy([0 fs/2],[1 1],'k:');
hold on;
    h = semilogy(f,abs(h).^2,'r');
    set(h,'LineWidth',0.4);
    hold off;
xlim([0 fs/2]);
xlabel('Frequency (Hz)');
set(get(gca,'YLabel'),'Interpreter','LaTeX');
ylabel('Magnitude Response $|H(e^{j\omega})|^2$');
title('Prediction Error Filter');
box off;
AxisSet(8);
print('NBIPredictionErrorFrequencyResponse','-depsc');
```

Example 1: Least Squares Filter MATLAB Code

```
function [c,yh] = LeastSquaresFilter(x,y,fsa,foa,wta,pfa);
%LeastSquaresFilter: Least squares estimate of FIR filter coefficients
%
%   [c,yh] = NonparametricSpectrogram(x,y,fs,wl,fr,nf,ns,pf);
%
%   x    Input signal.
%   y    Target signal.
%   fs   Sample rate (Hz). Default = 1 Hz.
%   fo   Filter order. Default = 10.
%   wt   Window type: 0=full (default), 1=none,
%        2=unbiased autocorrelation estimate
%   pf   Plot flag: 0=none (default), 1=screen, 2=current figure.
%
%   c    Vector of coefficients.
%   yh   Estimate of y using the estimator.
%
%   Calculates the least squares estimate of the coefficient vector
%   c using the input data. Efficiently calculates the
%   autocorrelation matrix using the recursive approach described
%   in Manolakis.
%
%   Example: Estimate the coefficients for doing narrowband
%   interfence of a microelectrode recording.
%
%      load MER.mat;
%      d    = round(2e-3*fs);
%      y    = x(d+(1:50e3));
%      x    = x(1:50e3);
%      [c,yh] = LeastSquaresFilter(x,y,fs,500,1,1);
%
%   D. G. Manolakis, V. K. Ingle, S. M. Kogon, "Statistical and
%   adaptive signal processing," Artech House, 2005.
%
%   Version 1.00 JM
```



```
%
%   See also signal, armcov, arcov, and aryule.

%==================================================================
% Error Checking
%==================================================================
if nargin<2,
    help LeastSquaresFilter;
    return;
    end;

if isempty(x) | isempty(y),
    error('Signal is empty.');
    end;

if length(x)~=length(y),
    error('Input signals are different lengths.');
    end;

if var(x)==0 | var(y)==0,
    error('Signal is constant.');
    end;

%==================================================================
% Calculate Basic Signal Statistics
%==================================================================
nx = length(x);                                             % No. samples in x
mx = mean(x);                                               % Input  signal mean
my = mean(y);                                               % Target signal mean

%==================================================================
% Process Function Arguments
%==================================================================
fs = 1;                                                     % Default sample rate
if exist('fsa','var') & ~isempty(fsa),
    fs = fsa;
    end;
```



```
fo = 10;                                                    % Default filter order
if exist('foa','var') & ~isempty(foa),
    fo = foa;
    end;

wt = 0;                                                     % Default filter order
if exist('wta','var') & ~isempty(wta),
    wt = wta;
    end;

pf = 0;                                                     % Default - no plotting
if nargout==0,                                              % Plot if no output arguments
    pf = 1;
    end;
if exist('pfa') & ~isempty(pfa),
    pf = pfa;
    end;

%==================================================================
% Preprocessing
%==================================================================
x  = x(:);                                                  % Make into a column vector
%x  = x - mx;                                               % Remove mean

%==================================================================
% Estimate the ACF
%==================================================================
if wt==0 | wt==2,
    np = 2^nextpow2(2*nx-1);                                % Figure out how much to pad the signal
    X  = fft(x,np);
    xc = ifft(abs(X).^2);
    ac = real(xc(1:fo));

    Y  = fft(y,np);
    cp = ifft(Y.*conj(X));
    cc = real(cp(1:fo));
```



```
    if wt==0,                                               % Biased estimate
        ac = ac./nx;
        cc = cc./nx;
    else                                                    % Unbiased estimate
        k  = (0:fo-1).';
        ac = ac./(nx-k);
        cc = cc./(nx-k);
        end;
    end;

%==================================================================
% Create the Estimated Autocorrelation (R) and Cross-Correlation (d)
%==================================================================
R = zeros(fo,fo);
d = zeros(fo,1);
switch wt,
case {0,2},
    for c1=1:fo,
        d(c1) = cc(c1);
        for c2=c1:fo,
            R(c1,c2) = ac(c2-c1+1);
            R(c2,c1) = R(c1,c2);
            end;
        end;
case 1,
    %--------------------------------------------------------------
    % Calculate the First Row
    %--------------------------------------------------------------
    for c1=1:fo,
        d(c1)   = sum(y(fo:nx).*x(fo-(c1-1):nx-(c1-1)));
        R(1,c1) = sum(x(fo:nx).*x(fo-(c1-1):nx-(c1-1)));
        R(c1,1) = R(1,c1);
        end;
    %--------------------------------------------------------------
    % Fill out the Remainder of the Matrix
    %--------------------------------------------------------------
    for c1=2:fo,
```

```
        for c2=c1:fo,
            R(c1,c2) = R(c1-1,c2-1) + x(fo-(c1-1)).*x(fo-(c2-1)) - x(nx-(c1-2)).*x(nx-(c2-2));
            R(c2,c1) = R(c1,c2);
            end;
        end;
        %X = zeros(nx-fo+1,fo);                              % Verificiation code
        %for c1=1:fo,
        %    X(:,c1) = x(fo-(c1-1):nx-(c1-1));
        %    end;
        %R2 = X'*X;
        %d2 = X'*y(fo:nx);
        %disp([max(max(abs(R-R2))) max(abs(d-d2))]);
    end;

%====================================================================
% Calculate the Coefficients
%====================================================================
c  = pinv(R)*d;
yh = filter(c,1,x);

%====================================================================
% Plot Results
%====================================================================
if pf>=1,
    if pf~=2,
        figure;
        end;
    FigureSet(1);

    k = 1:nx;
    t = (k-0.5)/fs;
    h = plot(t,y,'r',t,yh,'g');
    set(h,'LineWidth',1.2);
    xlim([0 nx/fs]);
    xlabel('Time (s)');
    ylabel('Signal');
    legend('Observed','Estimated');
```



```
    set(get(gca,'Title'),'Interpreter','LaTeX');
    title(sprintf('NMSE=%5.1f\\%%',100*sum((y-yh).^2)/sum((y-my).^2)));
    box off;
    AxisSet;

    if pf~=2,
        figure;
        end;
    FigureSet(2);
    [h,f] = freqz(c,1,2^12,fs);
    h = semilogy(f,abs(h).^2,'r');
    set(h,'LineWidth',1.2);
    xlim([0 fs/2]);
    xlabel('Frequency (Hz)');
    set(get(gca,'YLabel'),'Interpreter','LaTeX');
    ylabel('Magnitude Response $|H(\exp{j\omega})|^2$');
    box off;
    AxisSet;

    end;

%====================================================================
% Process Return Arguments
%====================================================================
if nargout==0,
    clear('c','yh');
    end;
```