12.6 Sequential LMMSE Estimation

Same kind if setting as for Sequential LS…

Fixed number of parameters (but here they are modeled as random)

Increasing number of data samples

Data Model: $\mathbf{x}[n] = \mathbf{H}[n]\boldsymbol{\theta} + \mathbf{w}[n]$

$(n+1)\times 1$
$\mathbf{x}[n] = [x[0] \ \ldots \ x[n]]^T$

$(n+1)\times p$
known

$\mathbf{H}[n] = \begin{bmatrix} \mathbf{H}[n-1] \\ \mathbf{h}^T[n] \end{bmatrix}$

$p\times 1$
unknown PDF
known mean & cov

$(n+1)\times 1$
$\mathbf{w}[n] = [w[0] \ \ldots \ w[n]]^T$
unknown PDF
known mean & cov
$\mathbf{C}_\mathbf{w}$ must be diagonal
with elements σ^2_n

$\boldsymbol{\theta}$ & $\mathbf{w}$ are uncorrelated

Goal: Given an estimate $\hat{\boldsymbol{\theta}}[n-1]$ based on $\mathbf{x}[n-1]$, when new data sample $x[n]$ arrives, update the estimate to $\hat{\boldsymbol{\theta}}[n]$

Development of Sequential LMMSE Estimate

Our Approach Here: Use vector space ideas to derive solution for "**DC Level in White Noise**" then write down general solution.

$$x[n] = A + w[n]$$

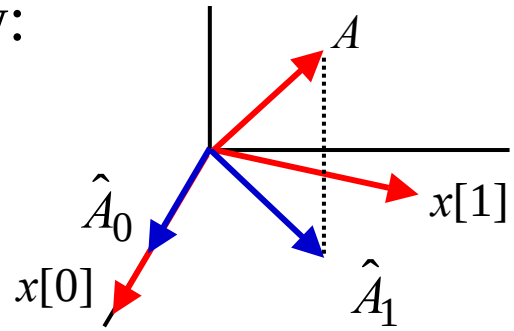
For convenience… Assume both A and $w[n]$ have zero mean

Given $x[0]$ we can find the LMMSE estimate

$$\hat{A}_0 = \left[\frac{E\{Ax[0]\}}{E\{x^2[0]\}}\right]x[0] = \left[\frac{E\{A(A+w[n])\}}{E\{(A+w[n])^2\}}\right]x[0] = \left[\frac{\sigma_A^2}{\sigma_A^2+\sigma^2}\right]x[0]$$

Now we seek to sequentially update this estimate with the info from $x[1]$…

- From Vector Space View:



- First project $x[1]$ onto $x[0]$ to get $\hat{x}[1|0]$

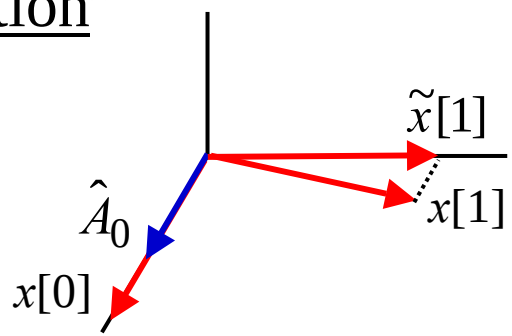
Estimate new data given old data… Prediction!

Notation: the estimate "at 1" based "on 0"

- Use Orthogonality Principle

$$\tilde{x}[1] \stackrel{\Delta}{=} x[1] - \hat{x}[1|0] \quad \text{is} \perp \text{to} \quad x[0]$$

$\Rightarrow$ This is the new, non-redundant info provided by data $x[1]$
It is called the "innovation"



- Find Estimation Update by Projecting A onto Innovation

$$\Delta\hat{A}_1 = \left\langle A, \frac{\tilde{x}[1]}{\|\tilde{x}[1]\|} \right\rangle \frac{\tilde{x}[1]}{\|\tilde{x}[1]\|} = \left\langle A, \frac{\tilde{x}[1]}{\|\tilde{x}[1]\|^2} \right\rangle \tilde{x}[1] = \left[\frac{E\{A\tilde{x}[1]\}}{E\{\tilde{x}^2[1]\}} \right] \tilde{x}[1]$$

Gain: k_1

- Recall Property: Two Estimates from $\perp$ data just add:

$$\tilde{x}[1] \perp x[0]$$

$$\hat{A}_1 = \hat{A}_0 + \Delta\hat{A}_1$$

$$= \hat{A}_0 + k_1\tilde{x}[1]$$

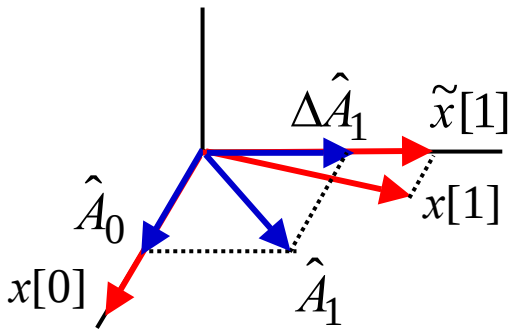
$$\hat{A}_1 = \hat{A}_0 + k_1\left[x[1] - \hat{x}[1|0]\right]$$

Old Estimate

Gain

New Data

Predicted New Data

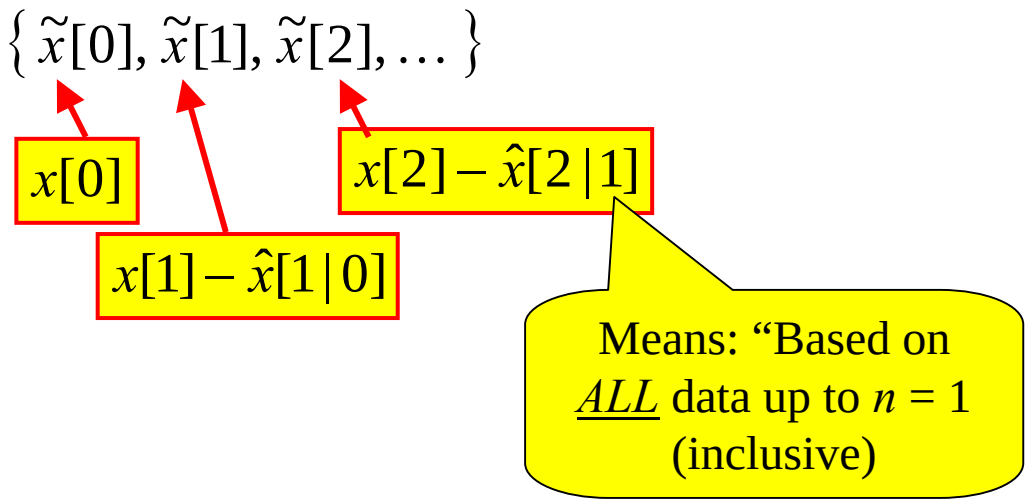


"Innovation" is $\perp$ Old Data

The Innovations Sequence

The Innovations Sequence is…

- Key to the derivation & implementation of Seq. LMMSE
- A sequence of orthogonal (i.e., uncorrelated) RVs
- Broadly significant in Signal Processing and Controls



General Sequential LMMSE Estimation

Initialization No Data Yet! $\Rightarrow$ Use Prior Information

$$\hat{\boldsymbol{\theta}}_{-1} = E\{\boldsymbol{\theta}\}$$ **Estimate**

$$\mathbf{M}_{-1} = \mathbf{C}_{\boldsymbol{\theta}\boldsymbol{\theta}}$$ **MMSE Matrix**

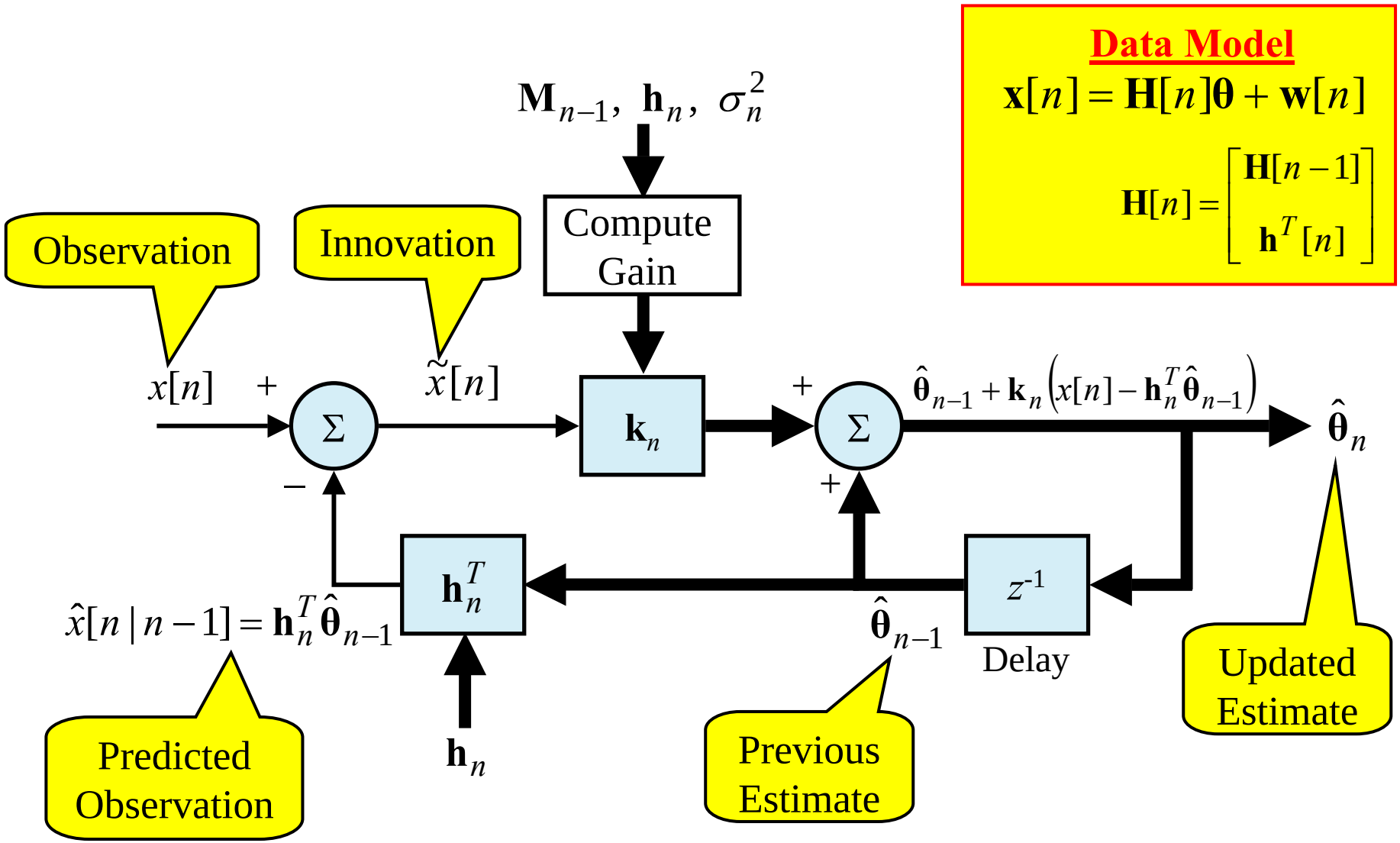
Update Loop For $n = 0, 1, 2, \ldots$

$$\mathbf{k}_n = \frac{\mathbf{M}_{n-1}\mathbf{h}_n}{\sigma_n^2 + \mathbf{h}_n^T \mathbf{M}_{n-1}\mathbf{h}_n}$$ **Gain Vector Calculation**

$$\hat{\boldsymbol{\theta}}_n = \hat{\boldsymbol{\theta}}_{n-1} + \mathbf{k}_n \left[x[n] - \mathbf{h}_n^T \hat{\boldsymbol{\theta}}_{n-1} \right]$$ **Estimate Update**

$$\mathbf{M}_n = \left[\mathbf{I} - \mathbf{k}_n \mathbf{h}_n^T \right] \mathbf{M}_{n-1}$$ **MMSE Matrix Update**

Sequential LMMSE Block Diagram



Exact Same Structure as for Sequential Linear LS!!

Comments on Sequential LMMSE Estimation

1. Same structure as for sequential linear LS. ***BUT***… they solve the estimation problem under very different assumptions.
2. No matrix inversion required… So computationally Efficient
3. Gain vector $\mathbf{k}_n$ weighs confidence in new data (σ^2_n) against all previous data ($\mathbf{M}_{n-1}$)
   - when previous data is better, gain is small… don't use new data much
   - when new data is better, gain is large… new data is heavily used
4. If you know noise statistics σ^2_n and observation rows $\mathbf{h}_n^T$ over the desired range of n:
   - Can run MMSE Matrix Recursion without data measurements!!!
   - This provides a Predictive Performance Analysis

12.7 Examples – Wiener Filtering

During WWII, Norbert Wiener developed the mathematical ideas that led to the Wiener filter when he was working on ways to improve anti-aircraft guns.

He posed the problem in C-T form and sought the best linear filter that would reduce the effect of noise in the observed A/C trajectory.

He modeled the aircraft motion as a wide-sense stationary random process and used the MMSE as the criterion for optimality. The solutions were not simple and there were many different ways of interpreting and casting the results.

The results were difficult for engineers of the time to understand.

Others (Kolmogorov, Hopf, Levinson, etc.) developed these ideas for the D-T case and various special cases.

Weiner Filter: Model and Problem Statement

Signal Model: $x[n] = s[n] + w[n]$

Noise
Model as WSS, Zero-Mean
$\mathbf{C}_{ww} = \mathbf{R}_{ww}$

Observed: Noisy Signal
Model as WSS, Zero-Mean
$\mathbf{C}_{xx} = \mathbf{R}_{xx}$

Desired Signal
Model as WSS, Zero-Mean
$\mathbf{C}_{ss} = \mathbf{R}_{ss}$

correlation matrix $\mathbf{R}_{xx} = E\{\mathbf{x}\mathbf{x}^T\}$

covariance matrix $\mathbf{C}_{xx} = E\{(\mathbf{x} - E\{\mathbf{x}\})(\mathbf{x} - E\{\mathbf{x}\})^T\}$

Same if zero-mean

Problem Statement: Process $x[n]$ using a linear filter to provide a "de-noised" version of the signal that has minimum MSE relative to the desired signal

LMMSE Problem!

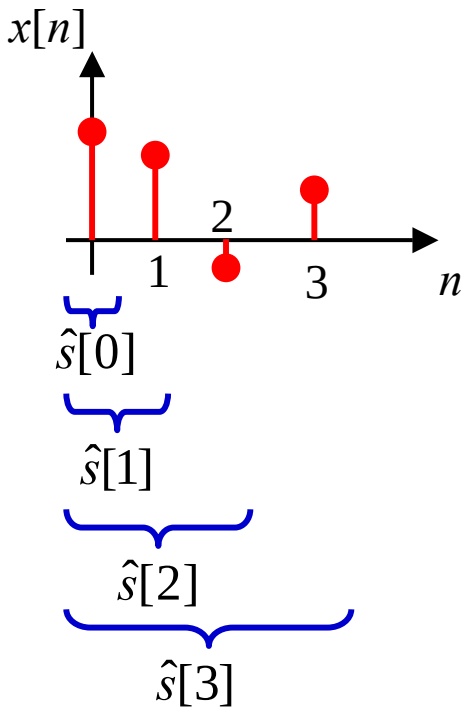
Filtering, Smoothing, Prediction

Terminology for three different ways to cast the Wiener filter problem

Filtering

Given: $x[0], x[1], \ldots, x[n]$

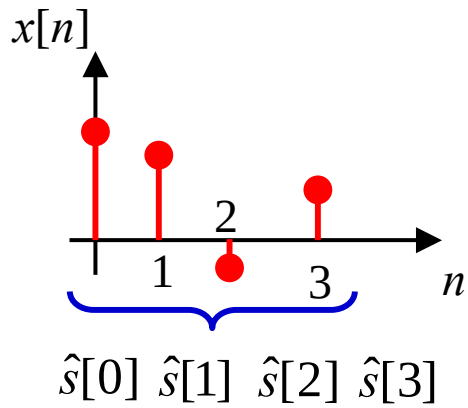
Find: $\hat{s}[n]$



Smoothing

Given: $x[0], x[1], \ldots, x[N-1]$

Find: $\hat{s}[0], \hat{s}[1], \ldots, \hat{s}[N-1]$

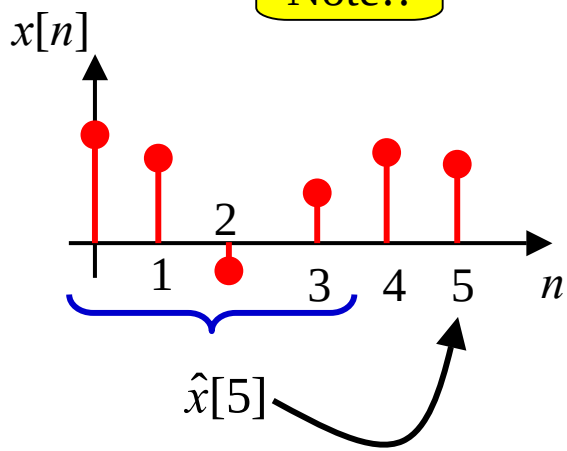


Prediction

Given: $x[0], x[1], \ldots, x[N-1]$

Find: $\hat{x}[N+l], \quad l > 0$

Note!!



All three solved using General LMMSE Est.

$$\hat{\boldsymbol{\theta}} = \mathbf{C}_{\boldsymbol{\theta}\mathbf{x}}\mathbf{C}_{\mathbf{xx}}^{-1}\mathbf{x}$$

$$\hat{\boldsymbol{\theta}} = \mathbf{C}_{\boldsymbol{\theta}\mathbf{x}}\mathbf{C}_{\mathbf{xx}}^{-1}\mathbf{x}$$

Filtering

$\theta = s[n]$ (scalar)

$$\mathbf{C}_{\theta\mathbf{x}} = E\left\{s[n]\mathbf{x}^T\right\}$$
$$= E\left\{s[n]\mathbf{s}^T\right\}$$
$$= \left[r_{ss}[n]\cdots r_{ss}[0]\right]$$
$$= \tilde{\mathbf{r}}_{ss}^{\mathrm{T}} \text{ (vector!)}$$

$$\mathbf{C}_{xx} = E\left\{(\mathbf{s}+\mathbf{w})(\mathbf{s}+\mathbf{w})^T\right\}$$
$$= E\left\{\mathbf{s}\mathbf{s}^T + \mathbf{w}\mathbf{w}^T\right\}$$
$$= \mathbf{R}_{ss} + \mathbf{R}_{ww}$$

$$\hat{s}[n] = \tilde{\mathbf{r}}_{ss}^T(\mathbf{R}_{ss}+\mathbf{R}_{ww})^{-1}\mathbf{x}$$
$$[1\times(n+1)][(n+1)\times(n+1)][(n+1)\times 1]$$

Smoothing

$\boldsymbol{\theta} = \mathbf{s}$ (vector)

$$\mathbf{C}_{\boldsymbol{\theta}\mathbf{x}} = E\left\{\mathbf{s}\mathbf{x}^T\right\}$$
$$= E\left\{\mathbf{s}(\mathbf{s}+\mathbf{w})^T\right\}$$
$$= E\left\{\mathbf{s}\mathbf{s}^T + \mathbf{s}\mathbf{w}^T\right\}$$
$$= \mathbf{R}_{ss} \text{ (Matrix!)}$$

$$\mathbf{C}_{xx} = E\left\{(\mathbf{s}+\mathbf{w})(\mathbf{s}+\mathbf{w})^T\right\}$$
$$= E\left\{\mathbf{s}\mathbf{s}^T + \mathbf{w}\mathbf{w}^T\right\}$$
$$= \mathbf{R}_{ss} + \mathbf{R}_{ww}$$

$$\hat{\mathbf{s}} = \mathbf{R}_{ss}(\mathbf{R}_{ss}+\mathbf{R}_{ww})^{-1}\mathbf{x}$$
$$[N\times N][N\times N][N\times 1]$$

Prediction

$\theta = x[N-1+l]$ (scalar)

***x* not *s*!**

$$\mathbf{C}_{\theta x} = E\left\{x[N-1+l]\mathbf{x}^T\right\}$$
$$= \left[r_{xx}[N-1+l]\cdots r_{xx}[l]\right]$$
$$= \tilde{\mathbf{r}}_{xx}^{\mathrm{T}} \text{ (vector!)}$$

$$\mathbf{C}_{xx} = E\left\{\mathbf{x}\mathbf{x}^T\right\}$$
$$= \mathbf{R}_{xx}$$

$$\hat{x}[N-1+l] = \tilde{\mathbf{r}}_{xx}^T\mathbf{R}_{xx}^{-1}\mathbf{x}$$
$$[1\times N][N\times N][N\times 1]$$

Comments on Filtering: FIR Wiener

$$\hat{s}[n] = \underbrace{\tilde{\mathbf{r}}_{ss}^{T}(\mathbf{R}_{ss}+\mathbf{R}_{ww})^{-1}}_{\mathbf{a}^T}\mathbf{x} = \mathbf{a}^T\mathbf{x}$$

$$\mathbf{h} = \begin{bmatrix} h^{(n)}[0] & h^{(n)}[1] & \dots & h^{(n)}[n] \end{bmatrix}^T = \begin{bmatrix} a_n & a_{n-1} & \dots & a_0 \end{bmatrix}^T$$

$$\hat{s}[n] = \sum_{k=0}^{n} h^{(n)}[k]\,x[n-k]$$

Wiener Filter as Time-Varying FIR Filter

- Causal!
- Length Grows!

Wiener-Hopf Filtering Equations

$$\underbrace{(\mathbf{R}_{ss}+\mathbf{R}_{ww})}_{\mathbf{R}_{xx}}\mathbf{h} = \mathbf{r}_{ss}$$

$$\mathbf{r}_{ss} = \begin{bmatrix} r_{ss}[0] & r_{ss}[1] & \dots & r_{ss}[n] \end{bmatrix}^T$$

$$\underbrace{\begin{bmatrix} r_{xx}[0] & r_{xx}[1] & \cdots & r_{xx}[n] \\ r_{xx}[1] & r_{xx}[0] & \cdots & r_{xx}[n-1] \\ \vdots & \vdots & \ddots & \vdots \\ r_{xx}[n] & r_{xx}[n-1] & \cdots & r_{xx}[0] \end{bmatrix}}_{\text{Symmetric \& Toeplitz}} \begin{bmatrix} h^{(n)}[0] \\ h^{(n)}[1] \\ \vdots \\ h^{(n)}[n] \end{bmatrix} = \begin{bmatrix} r_{ss}[0] \\ r_{ss}[1] \\ \vdots \\ r_{ss}[n] \end{bmatrix}$$

In Principle: Solve WHF Eqs for filter **h** at each n

In Practice: Use Levinson Recursion to Recursively Solve

Comments on Filtering: IIR Wiener

Can Show: as $n \to \infty$ Wiener filter becomes Time-Invariant

Thus: $h^{(n)}[k] \to h[k]$

Then the Wiener-Hopf Equations become:

$$\sum_{k=0}^{\infty} h[k] r_{xx}[l-k] = r_{ss}[l] \quad l = 0, 1, \ldots$$

and these are solved using so-called "Spectral Factorization"

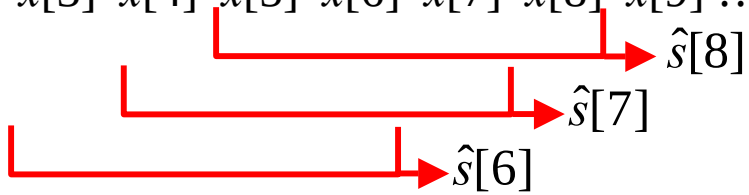
And… the Wiener Filter becomes IIR Time-Invariant:

$$\hat{s}[n] = \sum_{k=0}^{\infty} h[k] x[n-k]$$

Revisit the FIR Wiener: Fixed Length L

The way the Wiener filter was formulated above, the length of filter grew so that the current estimate was based on all the past data

Reformulate so that current estimate is based on only L most recent data: … $x[3]$ $x[4]$ $x[5]$ $x[6]$ $x[7]$ $x[8]$ $x[9]$ …



$$\hat{s}[n] = \sum_{k=0}^{L-1} h[k]x[n-k]$$

Wiener-Hopf Filtering Equations for WSS Process w/ Fixed FIR

$$\underbrace{(\mathbf{R}_{ss} + \mathbf{R}_{ww})}_{\mathbf{R}_{xx}}\mathbf{h} = \mathbf{r}_{ss}$$

$$\mathbf{r}_{ss} = \begin{bmatrix} r_{ss}[0] & r_{ss}[1] & \dots & r_{ss}[n] \end{bmatrix}^T$$

$$\underbrace{\begin{bmatrix} r_{xx}[0] & r_{xx}[1] & \cdots & r_{xx}[n] \\ r_{xx}[1] & r_{xx}[0] & \cdots & r_{xx}[n-1] \\ \vdots & \vdots & \ddots & \vdots \\ r_{xx}[n] & r_{xx}[n-1] & \cdots & r_{xx}[0] \end{bmatrix}}_{\text{Symmetric \& Toeplitz}} \begin{bmatrix} h[0] \\ h[1] \\ \vdots \\ h[n] \end{bmatrix} = \begin{bmatrix} r_{ss}[0] \\ r_{ss}[1] \\ \vdots \\ r_{ss}[n] \end{bmatrix}$$

Solve W-H Filtering Eqs ONCE for filter h

Comments on Smoothing: FIR Smoother

$$\hat{\mathbf{s}} = \underbrace{\mathbf{R}_{ss}(\mathbf{R}_{ss} + \mathbf{R}_{ww})^{-1}}_{\mathbf{W}}\mathbf{x} = \mathbf{W}\mathbf{x}$$

Each row of **W** like a FIR Filter
- Time-Varying
- Non-Causal!
- Block-Based

To interpret this – Consider N=1 Case:

$$\hat{s}[0] = \left[\frac{r_{ss}[0]}{r_{ss}[0] + r_{ww}[0]}\right]x[0] = \underbrace{\left[\frac{SNR}{SNR+1}\right]}_{\approx \begin{cases} 1, \text{ High SNR} \\ 0, \text{ Low SNR} \end{cases}} x[0]$$

Comments on Smoothing: IIR Smoother

Estimate $s[n]$ based on $\{\ldots, x[-1], x[0], x[1], \ldots\}$

$$\hat{s}[n] = \sum_{k=-\infty}^{\infty} h[k]x[n-k]$$

Time-Invariant & Non-Causal IIR Filter

The Wiener-Hopf Equations become:

$$\sum_{k=-\infty}^{\infty} h[k]r_{xx}[l-k] = r_{ss}[l] \quad -\infty < l < \infty$$

Differs From Filter Case
Sum over all k

Differs From Filter Case
Solve for all l

$$h[n] * r_{xx}[n] = r_{ss}[n]$$

$$H(f) = \frac{P_{ss}(f)}{P_{xx}(f)} = \frac{P_{ss}(f)}{P_{ss}(f) + P_{ww}(f)}$$

$H(f) \approx 1$ when $P_{ss}(f) >> P_{ww}(f)$
$H(f) \approx 0$ when $P_{ss}(f) << P_{ww}(f)$

Relationship of Prediction to AR Est. & Yule-Walker

Wiener-Hopf Prediction Equations

$$\mathbf{R}_{xx}\mathbf{h} = \mathbf{r}_{xx}$$

$$\mathbf{r}_{xx} = \begin{bmatrix} r_{xx}[l] & r_{xx}[l+1] & \ldots & r_{xx}[l+N-1] \end{bmatrix}^T$$

$$\underbrace{\begin{bmatrix} r_{xx}[0] & r_{xx}[1] & \cdots & r_{xx}[N-1] \\ r_{xx}[1] & r_{xx}[0] & \cdots & r_{xx}[N-2] \\ \vdots & \vdots & \ddots & \vdots \\ r_{xx}[N-1] & r_{xx}[N-2] & \cdots & r_{xx}[0] \end{bmatrix}}_{\text{Symmetric \& Toeplitz}} \begin{bmatrix} h[0] \\ h[1] \\ \vdots \\ h[n] \end{bmatrix} = \begin{bmatrix} r_{xx}[l] \\ r_{xx}[l+1] \\ \vdots \\ r_{xx}[l+N-1] \end{bmatrix}$$

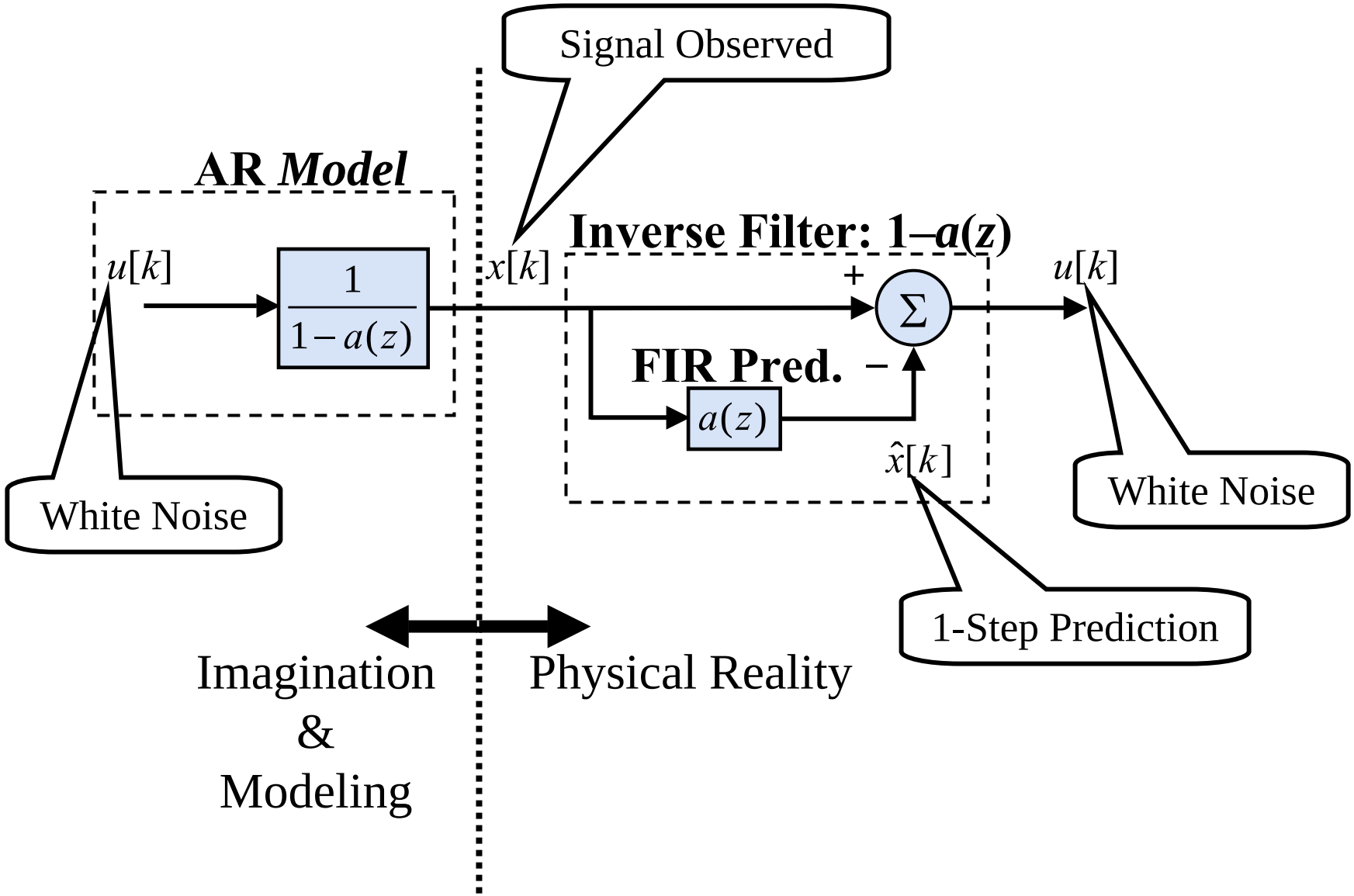
For l=1 we get EXACTLY the Yule-Walker Eqs used in Ex. 7.18 to solve for the ML estimates of the AR parameters!!

➔ **FIR Prediction Coefficients are estimated AR parms**

Recall: we first *estimated* the ACF lags $r_{xx}[k]$ using the data
Then used the estimates to find estimates of the AR parameters

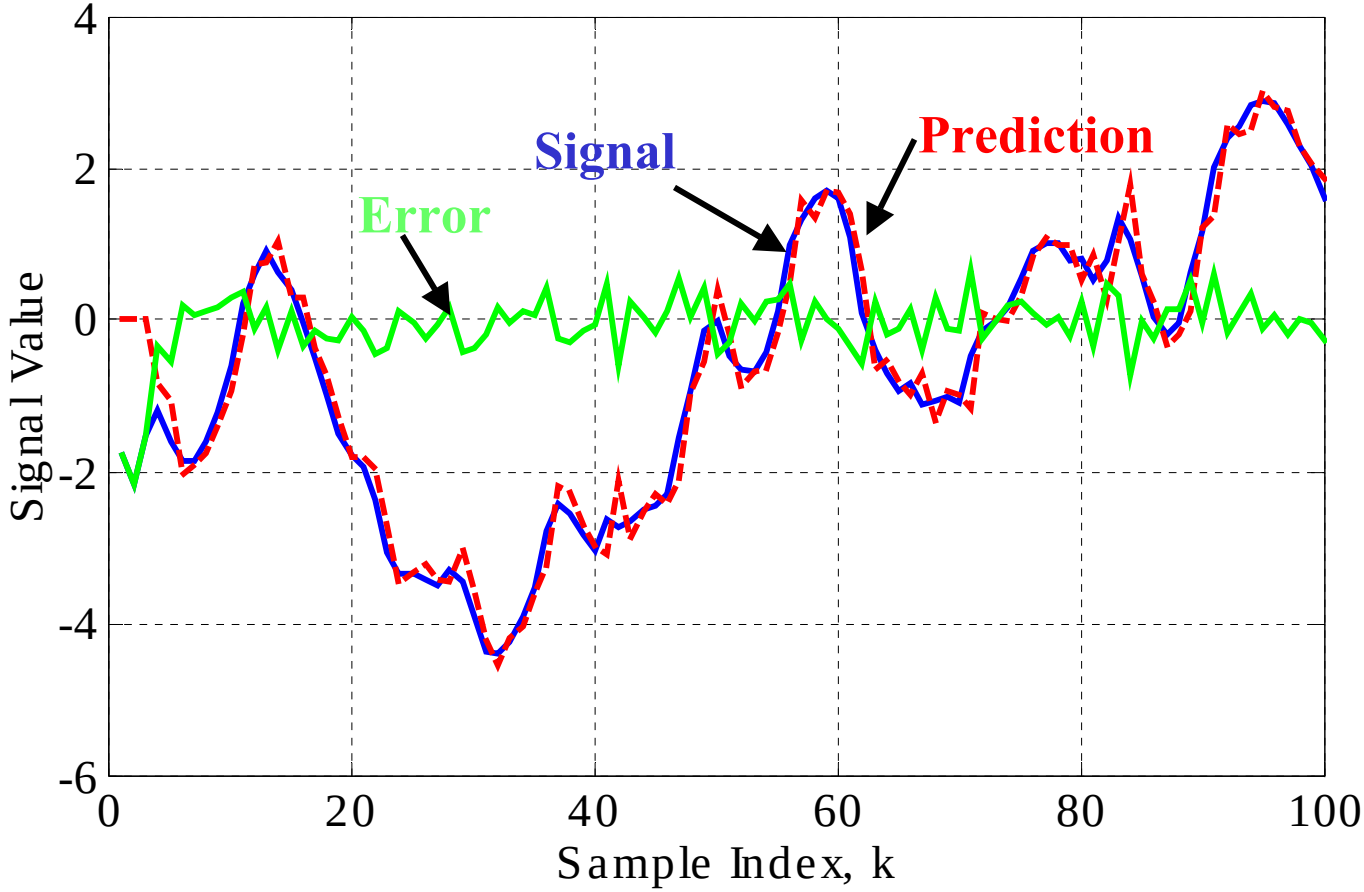
$$\hat{\mathbf{R}}_{xx}\mathbf{h} = \hat{\mathbf{r}}_{xx}$$

Relationship of Prediction to Inverse/Whitening Filter



Results for 1-Step Prediction: For AR(3)

At each k we predict $x[k]$ using past 3 samples



Application to Data Compression

Smaller Dynamic Range of Error gives More Efficient Binary Coding
(e.g., DPCM – Differential Pulse Code Modulation)