ECE 251B - Bhashar Rao (off-hours: Jacobs hall 6407) (Tue, Wed 1-9pm)
brao@ucsd.edu

Text:
1) Statistical and Adaptive Signal Processing
   Manolakis, Tugle, Kogon
2) Spectral Analysis of Signals
   Stoica & Moses (pdf on web)
3) Adaptive Filters
   Ali Sayed (lectures available, web-UCLA)

Topics:
1) Modeling of Random Processes
   a. Moving average, Auto Regressive (AR), ARMA
   b. Estimating model parameters, Levinson Durbin algo, Levinson
      Linear prediction & AR models (Linear Mean Squared Estimation),
      Backward prediction & Lattice filters.
2) Non-Stationary Signals
   - Kalman Filters
   - Adaptive Filters, LMS algo, Recursive Least Squares.

Grading: weekly hw - 30% (theoretic + Matlab) due Thurs
        being sheet, midterm exam - 30%
        of notes, Final exam - 50%
Analysis & Manipulation of Random Signals
models prediction, filtering, tracking

Random Process $x[n]$ - a collection/ensemble of sequences (possible realizations)

$(S, F, \mathbb{P})$ (field, probability measure)

Simple space (collection of subsets of $S$)
Set of outcomes (collection of subset of $S$) (events)

- Random Variable (r.v.): $x(n): S \rightarrow \mathbb{R}$
- Random Processes $x[n]: S \rightarrow$ sequence
  - if you fix $\omega$ and you have a realization: $x(n)$
  - if you fix $n$ - we have a r.v. of $x$ at time $n$
  - if fix $n$ and $\omega$ - we have a number.

- We'll usually mark a r.v. $x[n].$

ex: 1. $x[n] = A \cos(\omega_0 n + \phi), \quad \phi \sim U(-\pi, \pi)$
  - expected value over realizations: $E[x(n)] = 0.$

2. Gaussian white noise
   $x[n] = w[n]$
   $E(w[n]) = 0, \quad E(w[n]w[n]) = \delta[n], \delta[n]$
3. \( x[n] = ax[n-1] + w[n] \) \( |a| < 1 \)

\[
\frac{1}{1-az^{-1}} \quad \Rightarrow \quad x[n] \quad (1\text{st order AR process})
\]

4. \( x[n] = A \cdot \text{AR}(1) \) 
any realization is a constant, but the expected value at any time \( n \) is \( E(x[n]) = 0 \).

Wide sense stationary (WSS) process has simplifying assumptions:
1. \( E(x[n]) = \mu \) mean is constant 
2. \( E[x[n]x[n-m]] = \rho(m) \) autocorrelation is function of \( m \) 

Auto-covariance: \( E[(x[n]-\mu)(x[n-m]-\mu)] = \rho(m)-\mu^2 \)

Time averages (Ergodicity)
Using time averages to calculate ensemble averages is only valid for ergodic processes.
- Mean Ergodic: \( \hat{\mu}_n = \frac{1}{n} \sum_{n=0}^{n-1} x(n) \xrightarrow{n \to \infty} \mu \)

Mean squared convergence
\( E(\hat{\mu}_n - \mu)^2 \xrightarrow{n \to \infty} 0 \)

Examples that are mean-ergodic:
1) \((a)\), 2) \((\text{Gaussian w/o})\), 3) also.
4) is not mean ergodic - averaging over time we get the constant \( A \) and not the real mean - zero.
Back by Gardner: Cyclostationary Process

Correlation Ergodic:
\[ X^x[n\ell] = \frac{1}{N} \sum_{n=0}^{N-1} x[n] x^*[n-n_0] \xrightarrow{\text{ergodic}} R_x[n\ell] \]

This is harder to check.

Autocorrelation Sequence \( R_x[n\ell] = E[x^n]\) properties:

a. \( R_x[n\ell] = R_x[n*-n\ell] \) (Hermitian sequence)
b. \( R_x[n\ell] = E[|x[n]\|^2] \geq |R_x[n\ell]| \)
c. \( R_x[n\ell] \) is a positive-semi-definite sequence
\[ \sum_{j} \sum_{j} a_j^* R_x[n\ell-j] \geq 0 \quad \forall \alpha \]

This property is hard to verify in the time domain.

If we are given \( R_x[n\ell] \) for \( |n\ell| \leq N \) can we extend it to a valid correlation function (Maximum Entropy Method)

\[
X^T \begin{bmatrix} x[n-1] \\ x[n-2] \\ \vdots \\ x[n-M+1] \\ \vdots \\ x[n-M+1] \\ \vdots \\ x[n] \end{bmatrix} = R_m = E \begin{bmatrix} x_m[n-1] x^*_m[n] \\ x_m[n-2] x^*_m[n] \\ \vdots \\ x_m[n-M] x^*_m[n] \\ \vdots \\ x_m[n-M] x^*_m[n] \end{bmatrix}
\]

\[
\begin{bmatrix} E[x_0 x^*_0] & \cdots & E[x_0 x^*_N] \\ \vdots & \ddots & \vdots \\ E[x_N x^*_0] & \cdots & E[x_N x^*_N] \end{bmatrix} \]

This is a Toeplitz matrix.
1. It is also Hermitian: \( R_m = R_m^\dagger = (R_m^T)^\dagger \).
2. \( R_m \) is Toeplitz. The first row has all the required information.
3. \( R_m \) is a PSD matrix: \( R_m \succeq 0 \) for all \( a \)
   \[ a^\dagger R_m a \geq 0 \quad \forall a \]

HW1 is on dsp.ucsd.edu.
Rao will be out of town week 4 of May 26th.
Make-up sessions: Fri, 19th and 19th, 1-4-00 pm.

\[ r_{x[m]} = E(x[n+m]\cdot x^*[n]) \] is the autocorrelation function of a WSS process. We saw a properties of valid a'c:
1) Hermitian symmetry
2) \( r_{x[0]} = \| r_{x[m]} \| \)
3) PSD

If we are given \( r_{x[0]} = 1, r_{x[-1]} = r_{x[1]} = 0.9 \) can we fill \( r_{x[m]} = 0 \) \( \forall m \geq 2 \) and still have valid a'c? (for WSS's)

\[ \begin{align*}
\text{a'c matrix:} & \quad (\text{for WSS's}) \\
X_m[n] & = \begin{bmatrix} x[n] \\ \vdots \\ x[n-m+1] \end{bmatrix} \\
R_m & = E[ X_m[n] X_m^\dagger[n]\] \\
\text{this is Toeplitz matrix} & \begin{pmatrix}
\begin{bmatrix}
\begin{bmatrix} r[0] & r[1] \\ r[1] & r[2] \\ \vdots & \vdots \\ r[m-1] & r[m-2] \end{bmatrix}
\end{bmatrix}
\end{pmatrix}
\end{align*} \]
1) $R_m$ is Hermitian, $R_m = R_m^H$
2) $R_m$ is Toeplitz (first row contains all information)
3) PSD matrix $\mathbf{A}^*: \mathbf{A}^*R_mA \geq 0$.

If $M = 3$, we see that $R_m$ is not PSD, we can declare that the seq $r$ is not a valid ac seq.

$a^*R_ma$ is the power of output of an FIR filter

$y[n] = \sum_{k=0}^{\infty} h[k] x[n-k]$ is $[h[0] \ldots h[3]]^T [x[0] \ldots x[3]]^T$.

$E[|y[n]|^2] = h^*R_{\mathbf{A}^*}h$

$\mathbf{A}^*$ and $\mathbf{A}$ FIR filter $h$, this should be non-neg.

4) Eigenvalues of $R_m$ real (Hermitian)

Typically $\lambda_i \geq 0$ (Positive-Definite)

(we get $\lambda_i = 0$ when the process is predictable).

5) Trace ($\mathbf{R}_m$) Eigenvalue decomposition

$R_m = \mathbf{Q} \Lambda \mathbf{Q}^H = \begin{bmatrix} 9 & 9 & 9 & 9 \\ 9 & 9 & 9 & 9 \\ 9 & 9 & 9 & 9 \\ 9 & 9 & 9 & 9 \end{bmatrix} \begin{bmatrix} \lambda_1 & 0 & 0 & 0 \\ 0 & \lambda_2 & 0 & 0 \\ 0 & 0 & \lambda_3 & 0 \\ 0 & 0 & 0 & \lambda_4 \end{bmatrix} \mathbf{Q}^H$
and eigenvectors are orthonormal: $q_i^\mathsf{H} q_j = \delta_{ij}$

$Q^\mathsf{H} Q = I_m = Q Q^\mathsf{H}$

6) $\operatorname{trace}(R_m) = M \operatorname{trace}(E) = \sum_{i=1}^{M} \lambda_i$

$(\operatorname{tr}(AB) = \operatorname{tr}(BA))$ so

$= \operatorname{tr}(Q \Lambda Q^\mathsf{H}) = \operatorname{tr}(\Lambda Q^\mathsf{H} Q) = \operatorname{tr}(\Lambda) = \sum_{i=1}^{N} \lambda_i$

7) $\det(R_m) = \prod_{i=1}^{M} \lambda_i$

Let's examine the Fourier domain:

Power Spectral Density:

$R_x(e^{j\omega}) = \sum_{m=-\infty}^{\infty} R_x[m] e^{-jwm} = R_x(e^{j(w+2\pi)})$

$R_x[m] = \frac{1}{2\pi} \int_{-\pi}^{\pi} R_x(e^{j\omega}) e^{jwm} dw$

$\omega$-normalized freq. $\omega = \Omega T$

$R_x(z) = \sum_{m=-\infty}^{\infty} R_x[m] z^{-m}$  \hspace{1cm} $|z| < R_L$

includes unit circle

properties of Fourier transform $R_x(e^{j\omega})$:

1) $R_x(e^{j\omega}) = R_x^*(e^{j\omega})$. The power is real

(follows from hermitian of the time seq. $r[n]$)

2) $R_x(e^{j\omega}) = R_x(e^{-j\omega})$ (when $x[n]$ is real process)

3) $R_x(e^{j\omega}) \geq 0$ for $-\pi \leq \omega \leq \pi$ (consequence of psd $R_x[m]$)
we can check that power is non-neg

to validate $a^c x[n]$ is psd.

return to example $r_x[0] = 1$, $r_x[\pm 1] = 0.9$
and $r_x[\pm 2] = 0$. is it valid $a^c x[n]$?

$R_x(e^{j\omega}) = 1 + 1.8 \cos(\omega)$

this power is not always non-neg.
so zero expansion in this case is

not a valid expansion for $a^c x[n]$. 

3) $r_x[0] = \frac{1}{2\pi} \int_{-\pi}^{\pi} R_x(e^{j\omega}) \, dw \geq 0$

LTI & wss:

$x[n] \xrightarrow{H(z)} y[n]$

$y[n] = \sum_{k} h[k] x[n-k]$

typically causal systems: $y[n] = \sum_{k=0}^{\infty} h[k] x[n-k]$
typically assume stability $\sum_{k=0}^{\infty} |h[k]| < \infty$

$x[n]$ wss $\Rightarrow y[n]$ wss and

$x[n], y[n]$ are jointly wss.

cross-correlation is same of time $d/\Delta f$
\[
E(y[n]) = \sum \limits_{k} h[k] E(x[n-k]) = M_x \sum \limits_{k} h[k] = M_x H(z)
\]

\[
H(z) = \sum \limits_{n=-\infty}^{\infty} h[n] z^{-n} \quad |z| > r_R \quad \text{& includes unit circle}
\]

\[
H(e^{jw}) = H(2) \left|_{z = e^{jw}} \right.
\]

\[
\hat{r}_{xy}[m] = E[x[n+m] y[n]] = r_{yx}[z^{-m}]
\]

\[
= h^*[m] * r_x[m]
\]

\[
r_{yy}[m] = h[n] * r_{xy}[m] = h[n] * h^*[m] * r_{yx}[m]
\]

Cross spectral density:

\[
R_{xy}(e^{jw}) = \sum \limits_{m} r_{xy}[m] e^{j mw} = H^*(e^{jw}) R_x(e^{jw})
\]

\[
R_{xy}(z) = H^* \left( \frac{1}{z} \right) R_x(z)
\]

\[
= \frac{z \left[ h^*[m] \right]}{z \left[ h^*[m] \right]} \quad |z| < r_L
\]

\[
R_y(z) = H(z) H^* \left( \frac{1}{z} \right) R_x(z)
\]

\[
H(z) = 1 + (1+3i)z^{-1}
\]

\[
H^* \left( \frac{1}{z} \right) = 1 + (1-3i)z
\]

\[
R_y(e^{jw}) = \left| H(e^{jw}) \right|^2 R_x(e^{jw})
\]
\[ r_x(e^{j\omega}) = \frac{1}{2\pi} \int_{-\pi}^{\pi} R_x(e^{j\omega}) d\omega = \frac{1}{2\pi} \int_{-\pi}^{\pi} |H(e^{j\omega})|^2 R_x(e^{j\omega}) d\omega \]

\[ r_y(e^{j\omega}) = h^T R_a h^* \]

we can use this to show why \( R_x(e^{j\omega}) \geq 0 \)
otherwise we could find \( h^T \) that will cause \( r_y(e^{j\omega}) \leq 0 \).

\[ r_{y\bar{y}}(\bar{\omega}), r_{xy}(e^{j\omega}) \] can be applied to arbitrary jointly WSS processes, not necessarily \( x \) input & \( y \) output of LTI system.

\[ r_{y\bar{y}}(\bar{\omega}) = r_{yy}(e^{j\omega}) \]

coherence function \( G_{xy}(e^{j\omega}) = \frac{r_{xy}(e^{j\omega})}{\sqrt{r_{yy}(e^{j\omega})} \sqrt{r_{xx}(e^{j\omega})}} \)

some measure of predictability between \( x \) and \( y \).

Magnitude squared coherence \( |G_{xy}(e^{j\omega})|^2 = \frac{r_{xy}(e^{j\omega})^2}{r_{xx}(e^{j\omega}) r_{yy}(e^{j\omega})} \)

\( |G_{xy}(e^{j\omega})|^2 \leq 1 \). if it is close to 1, there is high predictability (for freq \( \omega \)).

\[ z[n] = x[n] + \alpha^* y[n] = (1, \alpha^*) [x[n]] [y[n]]^T \]

\[ r_{zz}[m] = E[z[n] z^*[n-m]] = E[(1, \alpha^*) [x[n]] [y[n]]^T] [x[n-m]]^T [y[n-m]]^T \]

\[ z[n] = z[n-m] \]
\[ \begin{align*}
&= (1, \alpha^*) \begin{bmatrix} r_x[m] & r_{xy}[m] \end{bmatrix} (1) \\
&= (1, \alpha^*) \begin{bmatrix} r_y[m] & r_{yx}[m] \end{bmatrix} (1)
\end{align*} \\
\]

\[ F_1: \]
\[ R_{xx}(e^{i\omega}) = (1, \alpha^*) \begin{bmatrix} R_x(e^{i\omega}) & R_{xy}(e^{i\omega}) & [1] \\
\text{non-neg} & \begin{bmatrix} R_{yx}(e^{i\omega}) & R_y(e^{i\omega}) \end{bmatrix} & \alpha \\
\text{so the matrix} & \begin{bmatrix} \alpha \end{bmatrix} \text{is psd} & \text{so the det of it is non neg:}
\]
\[ R_x(e^{i\omega})R_y(e^{i\omega}) - R_{xy}(e^{i\omega})R_{yx}(e^{i\omega}) \geq 0 \quad \forall \omega \]
\[ R_x(e^{i\omega})R_y(e^{i\omega}) - \left| R_{xy}(e^{i\omega}) \right|^2 \geq 0 \]
\[ \Rightarrow \frac{\left| R_{xy}(e^{i\omega}) \right|^2}{R_x(e^{i\omega})R_y(e^{i\omega})} \leq 1 \]

Modeling of Random Processes:
1) given \( r_{xx} \), generate \( x(w) \) that has \( r_{xx} \) as a/t func
2) Use models to do processing: prediction, noise cancellation...

Starting point: assume our observed process \( x(m) \)
is an output of LTI system with white noise input

\[ w[n] \rightarrow H(\omega) \rightarrow x[n] \]

zero mean
\[ \begin{bmatrix} w[n] & 0 \end{bmatrix} \]

By choosing \( H(\omega) \) we can try to match \( R_x(e^{i\omega}) \)
1. Uniqueness of the fitted model solution
2. How to choose one candidate?

\[ R_x(e^{j\omega}) = |H(e^{j\omega})|^2 \sigma_w^2 \]

We can choose \( \sigma_w^2 = 1 \)

The correlation doesn't have phase information, so solution can't be unique.

Innovations Representation

\[ \text{Linear Models} \]
\[ w[n] \xrightarrow{H(z)} x[n] \]

Assumed white Gaussian noise \( E(w[n]w[m]) = \delta[n-m] \sigma_w^2 \)

\[ R_x(e^{j\omega}) = \sigma_w^2 |H(e^{j\omega})|^2 \]
\[ R_x(z) = \sigma_w^2 H(z)H(z)^* \]
\[ x[n] = \sum_k h[k] w[n-k] \]

\[ r_x(m) = \sigma_w^2 h[m]h^*[m] = \sigma_w^2 \sum_k h[k]^* h[m+k] \]
\[ = \sigma_w^2 r_h[m] \]

*Given \( R_x(z)\), find \( H(z)\): spectral factorization

1) Paley Wiener condition:
\[ \int_{-\pi}^{\pi} \ln R_x(e^{i\omega}) \, d\omega < \infty \implies R_x(z) \text{ can be factored} \]

2) Solution is not unique: we can add an arbitrary all-pass filter \( wpj \xrightarrow{H(z)} H_{ap}(z) \xrightarrow{} X[n] \)

\[ H_{ap}(z) = \prod \frac{z - \frac{1}{a_k}}{z - \frac{1}{a_k}^{-1}} \] (just changes phase)

We'll not know the difference

3) Special factor. Innovations Model of factor \( H_I(z) \) is a special solution; we want both \( H_I(z) \) and \( \frac{1}{H_I(z)} \) to be stable & causal.

(requires both zeros and poles to be inside unit circle)

\[ W[n] \xrightarrow{H(z)} X[n] \xrightarrow{1}{H_I(z)} \xrightarrow{} W_{I}[n] \xrightarrow{H_I(z)} X[n] \]

This should have same statistics as \( wpj \)

we get \( R_x(e^{i\omega}) = \sigma_w^2 |H_I(e^{i\omega})|^2 \)

\[ X[n] = \sum_{k=0}^{\infty} h_I[k] W_I[n-k] \]
ex: \[ R_x(z) = \frac{P(z)}{Q(z)} \]
since \[ \lfloor x \rfloor [m] = x \lfloor x - m \rfloor \]
\[ R_x(z) = R_x(z^{-1}) \]

So \( z_i \) is zero \( \Rightarrow \frac{1}{z_i} \) is also a zero.
\( p \) pole \( \Rightarrow \frac{1}{p} \) is also pole.

\[ \frac{s_i}{z_i} \]

for stability, assign all poles inside unit circle to \( H(z) \).

Zero assignment is more flexible, but if we want the filter to also be invertible-stable causal we have to assign all the "inside"-zeros to \( H(z) \).

That's how we get innovation filter \( H_1(z) \).

Models: Difference Equations

\[ H(z) = \frac{B(z)}{A(z)} = \frac{\sum_{k=0}^{p} b_k z^{-k}}{1 + \sum_{k=1}^{\infty} a_k z^{-k}} \]

\[ X[n] = -\sum_{k=1}^{p} a_k x[n-k] + \sum_{k=0}^{\infty} w[n-k-j] \delta[k] \]

often \( d_0 = 1 \)
special cases:

1) \( A(\varepsilon) = 1 \). no feedback (no poles) in the system
\[ H(\varepsilon) = D(\varepsilon). \quad x[n] = w[n] + d_1 w[n-1] + \ldots + d_q w[n-q] \]

This is a moving average (MA) process.
Modeling involves choosing \( q \), free parameters and the values of the \( a_k \) parameters.

This is the same as assuming the correlation:
\[ r_x[n] = 0 \quad \text{for} \quad |m| > q \]
we just have to estimate \( r_x[0] \ldots r_x[q] \)
\[ \rightarrow R_x(\varepsilon) = \sum_{m=-q}^{q} r_x[m] \varepsilon^{-m} \]
and then do spectral factorization.
This method doesn't get much added value relative to Fourier methods, so for spectral estimation it is not so useful. But if the model itself is interesting, it might be useful.

2) \( D(\varepsilon) = 1 \), \( A(\varepsilon) = 1 + \sum_{k=1}^{P} a_k \varepsilon^{-k} \)
this is an AutoRegressive (AR) model of order \( P \).

\[ \begin{array}{c}
   w[n] \\
   \downarrow 1 \\
   A(\varepsilon) \\
   \rightarrow \\
   x[n]
\end{array} \]

This assumes \( x[n] \) has infinite autocorrelation
\[ x[n] = -\sum_{k=1}^{P} a_k x[n-k] + w[n] \]
Properties:

a. infinite extension

b. $a_1, \ldots, a_p, \sigma_w^2$ (p+1) parameters

c. $p$ is not that critical.

\[ R_x(e^{j\omega}) = \frac{\sigma_w^2}{|A(e^{j\omega})|^2} \]

Finding the parameters:

\[ x[n] = \begin{bmatrix} x[n-1] \\ \vdots \\ x[n-p] \end{bmatrix} \begin{bmatrix} a_1 \\ \vdots \\ a_p \end{bmatrix} + \begin{bmatrix} w[n] \end{bmatrix} \]

\[ = X_p[n-p] \begin{bmatrix} a_1 \\ \vdots \\ a_p \end{bmatrix} \]

\[ E \left[ x[n] w[n]^* \right] = 0 \quad (\text{line } n) \]

notice: $E \left[ x[n-L] w[n]^* \right] = 0 \quad (\text{line } n)$

depends on previous values of $w$, but not on $w[n]$.

So:

\[ X_p[n-p] = \begin{bmatrix} x[n-p] \\ \vdots \\ x[n-L] \end{bmatrix} \]

\[ \begin{bmatrix} x[n-1] \\ \vdots \\ x[n-p] \end{bmatrix} \begin{bmatrix} \sigma^2 \\ \vdots \\ \sigma^2 \end{bmatrix} = -E \left[ X_p[n-p] X_p^*[n-p] \right] \cdot a + E \left[ X_p[n-p] w[n]^* \right] \]

\[ \begin{bmatrix} r_x(1) \\ \vdots \\ r_x(p) \end{bmatrix} = -E \left[ X_p[n-p] X_p^*[n-p] \right] \cdot a \]

\[ R_p a = - \begin{bmatrix} r_x(1) \\ \vdots \\ r_x(p) \end{bmatrix} \begin{bmatrix} \sigma^2 \\ \vdots \\ \sigma^2 \end{bmatrix} \quad \text{Yule-Walker Equations} \]
we need to calc values $r_x(0), \ldots, r_x(p)$ to form the equation, and then solve it.

$$R_p = \begin{bmatrix} r_x(0) & r_x(1) & \cdots & r_x(p) \\ r_x(1) & r_x(2) & \cdots & r_x(p+1) \\ \vdots & \vdots & \ddots & \vdots \\ r_x(p) & r_x(p+1) & \cdots & r_x(2p) \end{bmatrix}$$

(the hermitian, Toeplitz matrix) that we know.

Let's look at $W(n)$ by the model:

$$W(n) = [\alpha_1, \alpha_2, \ldots, \alpha_p] X_{p+1}[n]$$

$$E[W(n)^* W(n)] = (\alpha, \alpha^*) R_{p+1}[a^*] = \sigma_w^2 \alpha$$

because $\sigma_w^2$ is real, positive.

$$W(n) \rightarrow \frac{1}{\sqrt{\alpha}} \rightarrow X(n) \rightarrow \frac{A(n)}{\sqrt{\alpha}} \rightarrow W(n)$$

Yule-Walker equations: \(1 \) $R_{p+1}[a] \rightarrow \frac{r_p}{r_0}$ \(2 \) $\sigma_w^2 = (\alpha, \alpha^*) R_{p+1}[a]$ Combine equation for $\alpha$ with the eq. for $\sigma_w^2$.

$$R_{p+1}[a] = \begin{bmatrix} \sigma_w^2 \\ 0 \end{bmatrix} \text{ extended Yule-Walker eq.}$$

Toepplitz matrix nesting:

$$R_{p+1} = \begin{bmatrix} r(0) & r(1) & \cdots & r(p) \\ r(1) & r(2) & \cdots & r(p+1) \\ \vdots & \vdots & \ddots & \vdots \\ r(p) & r(p+1) & \cdots & r(2p) \end{bmatrix} = \begin{bmatrix} R_p & \ldots & \vdots \\ \vdots & \ddots & \vdots \\ \vdots & \vdots & \ddots \\ r(p+1) & \ldots & r(2p) \end{bmatrix}$$
3) More general: ARMA \( H(z) = \frac{D(z)}{A(z)} \)

\[ w(n) \rightarrow \left[ \frac{D(z)}{A(z)} \right] \rightarrow x(n) \rightarrow \left[ \frac{A(z)}{} \right] \rightarrow \hat{y}(n) \]

(should be an MA process)

\[ x[n] = -\sum_{k=1}^{p} a_k x[n-k] + \sum_{k=0}^{q} b_k w[n-k] \]

\[ = -x[n-1]a + \sum_{k=0}^{q} b_k \left[ \sum_{n=0}^{w[n-k]} \right] \]

\[ E[x[n-l]w[n]] = 0 \quad \text{for } l > a \]

---

I missed start of Linear Mean Squared Estimate.

\( r\times X = [x_1, \ldots, x_r] \) is given. We want to estimate \( y(r) \) using linear estimate (scalar)

\[ \hat{y} = C^H x \]

Error = \( \hat{y} = y - \hat{y} = y - C^H x \)

\[ \text{error (var) } = \text{error (cov)} = \text{error (bias)} \]

\[ C_o = \text{arg min} P(c) = E \| y - \hat{y} \|^2 \]

we saw the solution satisfies:

\[ R_{xx} C_o = R_{yx} \]

and then:

\[ P_0 = P(C_o) = E(y - \hat{y})^2 = R_{xx} - R_{yx} R_{xx}^{-1} R_{yx} \]
\[ E[|Y|^2] = R_{yy}x_0 \]
\[ = E[|Y|^2] - C_0^\mathbb{H} R_{xy} \]
\[ y_{\text{rms}} = y_0 = C_0^\mathbb{H} x \]

Smallest \( y_{\text{rms}} \) error: \( y_{\text{rms}} = y_0 = y - C_0^\mathbb{H} x \)

Combine 2 equations (for params \( C_0 \) and for error power)

\[
\begin{bmatrix}
R_{xy} & R_{xx}
\end{bmatrix}
\begin{bmatrix}
1
\end{bmatrix} = C_0
\]
\[
\begin{bmatrix}
E[|Y|^2] & R_{xy}
\end{bmatrix}
\begin{bmatrix}
1
\end{bmatrix} = P_0
\]

\[
E \left[ y \mathbb{H} y x^{\mathbb{H}} \right]
\]

Now try to predict a vector \( y \) \( (d \times 1) \)
Now \( C_0 \) is a matrix \( m \times l \)
\( C_0 \) column is to predict \( y_{(i)} \)
Now the error \( \hat{y} \) is a vector \( (l \times 1) \)
\( P(c) = E \| \hat{y} \|^2 = \sum_{i=1}^{l} E |y_i|^2 \)

Each entry in \( P \) the sum depends just on one column
The solution will be:
\[ R_{xx} C_0 = R_{xy} \] \((m \times 2)\)
\[ P(C_0) = E \| y \|^2 - \text{tr}(R_{xx} R_{yy}^{-1} R_{xy}) \]
\[(E_1 1 y_1 1^2 = \text{tr}(E(y y^H)))\]

so the solution is equivalent to \( L \) separate MSE predictions of scalars \( y_i \).

Computation:
\[R_{xx} \text{ is hermitian} \quad R_{xx} = LL^H \quad (L \text{ lower triangular})\]
\[LL^H c = R_{xy}\]
\[\]  \[1) \quad LY = R_{xy} \rightarrow \text{solve for } L\]
\[2) \quad L^H c_0 = y \rightarrow \text{solve for } c_0\]

Orthogonality Property:
The error \( \hat{y}_{\text{rms}} \) is orthogonal to any linear combination of \( x \):
\[E(\hat{y}_{\text{rms}} X^H b^H) = 0\]

Proof:
\[E(\hat{y}_{\text{rms}} X^H b^H) = E(\hat{y}_{\text{rms}} X^H) B^H =\]
\[= E((y - c_0 X^H)^T X^H) B^H = (R_{yx} - c_0^H R_{xx}) B^H = 0\]
\[(\text{we know } R_{xx} c_0 = R_{yx} \rightarrow c_0^H R_{xx} = R_{yx})\]

Special cases:
1) \( B = I \) \( \hat{y}_{\text{rms}} \perp X \)
2) \( B = c_0^H \) \( \hat{y}_{\text{rms}} \perp \hat{y}_{\text{rms}} \)
An interesting manipulation:
\[ E(\tilde{Y}_{\text{LMS}} \tilde{Y}^*) = E(\tilde{Y}_{\text{LMS}}(\tilde{Y} - \tilde{Y}_{\text{LMS}} + \tilde{Y}_{\text{LMS}})^*) = \]
\[ = E(\tilde{Y}_{\text{LMS}}^2) + \sigma = P_0. \]

We can use orthogonality principle and find the solution to LMS:

Find \( c \) s.t. \( y^* = y - c^* x \perp Bx \\perp B \)

\[ x[n] = 0.9x[n-1] + w[n] \]
suppose we observe to previous values \( x = \begin{bmatrix} x[n-1] \\ x[n-2] \end{bmatrix} \)
we want to predict \( y = x[n] \)

We could solve the equations...

but we can also guess the best estimate should be \( \hat{y} = 0.9x[n-1] \)

Then we can prove this is indeed the optimal estimation.

\( y - \hat{y} = x[n] - 0.9x[n-1] = w[n] \perp x[n-2] \quad \forall n \geq 1 \)

we have error orthogonal to all observations.

\[ x[n] = 0.2x[n-1] + 0.9x[n-2] + w[n] \]

Best Affine Estimates:
so far we have \( \hat{y}_0 = C^H x \)
we want to have an unbiased estimator:

\[ E(\hat{y}_0) = c^H E(x) \quad \text{not necessarily equal to } E(y) \]

Affine class takes care of it - add constant.
\( \hat{Y} = C^H x + D \) now we'll deal with covariances instead of correlations.

Minimize:

\( P(C,D) = E(\hat{Y} - Y)^2 \)

The solution will be:

\[
\begin{align*}
\Sigma &= \Sigma_y - C_\Sigma x^x \\
C &= \Sigma_x^x (\Sigma_x \Sigma_y)^{\text{cross}} (\Sigma_x \Sigma_y)_{\text{covariance matrices}} \\

\end{align*}
\]

\( \Gamma_{xx} = E[(x-m_x)(x-m_x)^H] \)

\( \Gamma_{xy} = E[(x-m_x)(y-m_y)^H] \)

For vs's processes it's easier to get \( E(\hat{Y}) = my \), subtract it...

Nonlinear Estimation:

\( \hat{Y} = \phi(x) \)

\[
\begin{align*}
\min_{\phi} E(|Y - \hat{Y}|^2) \\

\end{align*}
\]

Best nonlinear estimate \( \hat{Y}_{ns} \) turns out to be:

\( \hat{Y}_{ns} = E(Y|x) \)

To compute we need \( f(y|x) \) mean and covariance are not enough.

Affine:

\( \hat{Y}_{ns} = \hat{Y}_{ns} \) when \( X,Y \) jointly Gaussian

(mean and covariances are enough)
Application to Time Series:

1) Prediction (observe \[x(n)\] and predict \[x(n+q)\] \(q \geq 1\))
   \[x(n) = d(n) + v(n)\]
   - \(d(n)\): desired signal
   - \(v(n)\): additive noise

   a) predict \(d(n)\) given \(x(n-l)\), \(l \geq 1\)
   b) filtering: predict \(d(n)\) given \(x(n-l)\), \(l \geq 0\)
   c) smoothing: predict \(d(n)\) given \(x(m)\) available

   (non-causal operation, you can look at the future)

   Fixed-lag smoothing uses a fixed length look at the future for smoothing delay will be small.

Applications:

1. Communications (equalizers)
2. Interference Suppression (remove corrupting noise)
3. Echo cancellations
4. Data compression (removing redundancy)
5. Geophysics, Radar & Sonar
6. Noise cancellation (when having extra data about the noise)
   \[x(n) = d(n) + v(n)\]
   \[w(n)\] is correlated with \(v(n)\) and not with \(d(n)\)

   \[
   \begin{array}{c}
   x(n) \\
   \downarrow \\
   0 \\
   \downarrow \\
   d(n) \\
   \hline
   \end{array}
   \]

   \[
   \begin{array}{c}
   w(n) \\
   \downarrow \\
   H(z) \\
   \hline
   v(n) \quad \text{Widrow: Adaptive noise cancelling}
   \end{array}
   \]
Let's summarise from last class:

\[ Y_{\text{lm}} = C_0^H x \]

\[ R_{xx} C_0 = R_{xy} \]

\[ \hat{Y}_{\text{lm}} = \hat{Y} - Y_{\text{lm}} \perp B x \]

\[ \hat{p}_o = E\left(\hat{Y}_{\text{lm}}^2\right) = R_{yy} - R_{yx} R_{xx}^{-1} R_{xy} \]

\[ = E(\hat{Y}_{\text{lm}} Y^*) \]

In the vector case, \( C \) will be a matrix.

Affine estimate - deal with non-zero mean:

\[ \hat{Y} = C^* x + D \]

Apply this framework to time series:

\[ \text{WSS } \times [n] \]

\[ x[n] = - \left( \sum_{k=1}^{n} a_k x[n-k] \right) \quad \text{(''c_o'' = -a)} \]

\[ e[n] = \hat{x}[n] - x[n] = [1, a_m] \hat{x}_{m+1}[n] \]

\[ \hat{X}^*[n] = X_{m+1}[n] \begin{bmatrix} 1 \\ a_m \end{bmatrix} / \text{multiply by } x_{m+1}[n] \]

and take expectation:

\[ E[\hat{x}_{m+1}[n] \hat{X}^*[n]] = E[x_{m+1}[n] X_{m+1}[n] \begin{bmatrix} 1 \\ a_m \end{bmatrix}] \]

\[ \text{P}_o = E[e[n]^2] = E(e[n]^* x[n]) \]
we get:
\[ \begin{bmatrix} P_0 \\ \sigma_m^2 \end{bmatrix} = R_{m+1} \begin{bmatrix} 1 \\ \alpha_m \end{bmatrix} \]
 forward linear prediction
 for any WSS process \( x[n] \)

Connection to AR Modeling:
suppose we have an AR model

\[ x[n] = - \sum_{k=1}^{P} \hat{\alpha}_k x[n-k] + w[n] \]

\[ R_{pp} \begin{bmatrix} 1 \\ \hat{\alpha}_p \end{bmatrix} = \begin{bmatrix} \sigma_w^2 \\ 0 \end{bmatrix} \] (Yule-Walker equations)

look the same as the linear prediction equations,
so if the process is indeed AR process, \( p \)-order
and we take \( M \geq p \)

\[ \hat{\alpha}_m = \begin{bmatrix} \hat{\alpha}_p^T \\ 0_{M-p} \end{bmatrix} \] we'd get \( \hat{P}_0 = \sigma_w^2 \)

we can try different \( M \)'s until we start getting zeros in coefficients - then it means we reached the original order of the AR process.

---

**Backward Prediction:**

\[ x[n-M] = - \sum_{k=0}^{M-1} b_k^* x[n-k] \]
linear combination of \( M \) samples ahead of the predicted one.

**backward prediction error:**

\[ e_b[n] = x[n-M] - x[n-M] = (b_m^T, 1) x[n] \]

\[ e_b[n] = x[n+1] \begin{bmatrix} b_m^T \\ 1 \end{bmatrix} \]

mult by \( x[n+1] \) and take expectation


\[ E(x_{m+1}^* e^{b^*}) = E(x_{m+1}^* x_{m+1}^H [b_m]) \]

\[
\begin{bmatrix}
0_m \\
p_c b
\end{bmatrix} = R_{m+1} \begin{bmatrix} b_m \\
p_c b
\end{bmatrix}
\]

backward linear prediction

\[
\begin{align*}
C_m[n] &= x[n] + \sum_{k=1}^{\infty} a_k^* x[n-k] \\
C_m^b[n] &= x[n-m] + \sum_{k=0}^{N-1} b_k^* x[n-k]
\end{align*}
\]

\[
R_{m+2} \begin{bmatrix} a_m \\
p_c a_m
\end{bmatrix} = R_m \begin{bmatrix} a_m \\
p_c a_m
\end{bmatrix} \quad \text{("Normal equations")}
\]

\[
R_{m+3} \begin{bmatrix} b_m \\
p_c b_m
\end{bmatrix} = \begin{bmatrix} C_m \\
p_c b_m
\end{bmatrix}
\]

4) \( p_m^f = p_m^b \) (positive real number, both directions get same error (same predictability power).

5) \( a_m = J b_m^* \). The coefficients are reverse-complex.

\((J = (0, i))\) - reverse operator

To prove lightly:

\[
R_{m+4} = J R_{m+1} R_{m+1} J \quad \text{(central hermitian)}
\]

we get: (start from backward equation)

\[
J R_{m+1} R_{m+1} J = \begin{bmatrix} 0_m \\
p_c b_m
\end{bmatrix} \Rightarrow J R_{m+1} R_{m+1} J \begin{bmatrix} b_m \\
p_c b_m
\end{bmatrix} = \begin{bmatrix} 0_m \\
p_c b_m
\end{bmatrix}
\]

\[
J R_{m+1} R_{m+1} J = \begin{bmatrix} 0_m \\
p_c b_m
\end{bmatrix} \Rightarrow J R_{m+1} R_{m+1} J = \begin{bmatrix} 0_m \\
p_c b_m
\end{bmatrix}
\]

\( J = J^{-1} \)

\[
= \text{Raat1 Jnt1} \begin{bmatrix} b^x_n \\ a_n \end{bmatrix} = \text{Jnt1} \begin{bmatrix} \sigma^2_m \\ a_n \end{bmatrix} \Rightarrow \text{Raat1} \begin{bmatrix} J_m b^x_n \\ a_n \end{bmatrix} = \begin{bmatrix} P_m^b \\ a_n \end{bmatrix}
\]

This is the forward pred equation. So:

- \( P_m^b \) must be \( P_m^f \)
- \( J_m b^x_n \) must be \( a_m \)

So we can use \( P_m \) (don't have to mark for \( b \))

*Extension to smoothing is readily possible.*

Think of the process as filtering:

\[
\begin{array}{c}
\hat{x}(n) \\
\xrightarrow{A_m(z)} \\
e_m(n)
\end{array}
\]

\[
A_m(z) = 1 + \sum_{k=2}^{M} a_k^* z^{-k}
\]

Can be viewed as a whitening process.

Can I do \( e_m(n) \xrightarrow{A_m(z)} \hat{x}(n) \)?

We need to \( A_m(z) \) must have its zeros inside unit circle (minimum phase filter) so that it will be invertible-stable-causal. \( |z| < 1 \)

\[
A_m(z) = \prod_{k=1}^{M} (1 - z_k z^{-1})
\]

3) \( A_m(z) \) is actually minimum phase.
Proof: (there are many different proofs)

\[ A_n(z) = G(z) \left( 1 - z_p z^{-1} \right) \]

\( G(z) \) has \( M-1 \) zeros, single zero

\[ X[n] \xrightarrow{\text{filter}} S[n] \]

We already know \( A(z) \) is optimal filter:

\[ e_f[n] \perp B X_{n-1} \]

\[ e_f[n] = S[n] - z_p S[n-1] \]

\[ \begin{align*}
& \text{(depends on} \quad \begin{bmatrix} X[n] \\ X[n-1] \end{bmatrix} \\
& \text{depen on} \quad \begin{bmatrix} X[n-1] \\ X[n-m] \end{bmatrix} 
\end{align*} \]

\[ e_f[n] \perp S[n-1] \]

mult by \( S[n-1] \) and take expectation:

\[ \mathbf{0} = r_0(\theta) - z_p r_\phi(\theta) \]

\[ z_p = r_\phi(\theta) \]

we know \( r_\phi(\theta) \leq r_\phi(\theta) \) so \( |z_p| < 1 \)

and \( \phi \) so if \( p_\phi > 0 \) \( |z_p| < 1 \)

\[ P_m = E\left[ |e_f[n]|^2 \right] = E\left[ |e_f[n]|^2 | e_f[n] \right] = E\left( \mathbf{A}^T \mathbf{A} (S[n] - z_p S[n-1]) \right) = \]

\[ = E\left( e_f[n]^T S[n] \right) = E\left( (S[n] - z_p S[n-1]) S[n] \right) = \]

\[ = r_\phi(\theta) - z_p^* r_\phi(\theta) = r_\phi(\theta) \left( 1 - |z_p|^2 \right) \]

so if \( p_\phi > 0 \) \( |z_p| < 1 \)

process is not perfectly predictable.

(always get some error).
Levinson-Durbin Algorithm (LDA)

Direct matrix inversion: complexity of $O(m^3)$

Inverting $R_m$ to solve for coefficients.

LDA: complexity $O(m^3)$

Using $a_m = \mathbf{J} b_m^*$ (using central hermitian property of $R_{m+1}$)

Using reciprocity property of $R_{m+1}$

Autoregressive process: computes $\frac{a_0}{p_0} \rightarrow \frac{a_1}{p_1} \rightarrow \frac{a_2}{p_2} \rightarrow \ldots$

Assume we know $a_m, p_m$ → we want to get $a_{m+1}, p_{m+1}$

We assume already solving $\begin{bmatrix} a_m \\ 1 \end{bmatrix} = \begin{bmatrix} \mathbf{J} b_m \\ p_m \end{bmatrix}$

and $\begin{bmatrix} b_m \\ 1 \end{bmatrix} = \begin{bmatrix} \mathbf{J} a_m \\ p_m \end{bmatrix}$.

We want to solve with $R_{m+2}$

Start with "initial guess":

$R_{m+2} \begin{bmatrix} a_m \\ 1 \end{bmatrix} = \begin{bmatrix} R_{m+1} \begin{bmatrix} a_m \\ 1 \end{bmatrix} + \begin{bmatrix} \mathbf{J} b_m \\ p_m \end{bmatrix} \begin{bmatrix} b_m \\ 1 \end{bmatrix} \\ \begin{bmatrix} \mathbf{J} a_m \\ p_m \end{bmatrix} \end{bmatrix}$

$\begin{bmatrix} a_m \\ 1 \end{bmatrix} = \begin{bmatrix} \mathbf{J} b_m \\ p_m \end{bmatrix}$

$\begin{bmatrix} \mathbf{J} b_m \\ p_m \end{bmatrix}$ is easily calculated.

If we see $b_m = 0$, we're lucky and we found the $(m+1)$ extension.

Do it for backward!

$R_{m+2} \begin{bmatrix} b_m \\ 1 \end{bmatrix} = \begin{bmatrix} \mathbf{J} a_m \\ p_m \end{bmatrix}$

$\begin{bmatrix} \mathbf{J} a_m \\ p_m \end{bmatrix}$
trick (take a linear combination):
\[
R_{m+2} \begin{bmatrix} a_m \\ 0 \\ b_m \end{bmatrix} = \begin{bmatrix} P_m \\ 0 \\ P_m \end{bmatrix} = \begin{bmatrix} \frac{P_m}{P_m^*} \\ 0 \\ \frac{P_m}{P_m^*} \end{bmatrix}
\]

we select \( k_m = -\frac{P_m}{P_m^*} \) to get the result we want.

recipe: after calculating \( \beta_m \) and \( k_m \):
\[
\begin{bmatrix} a_{m+1} \\ 0 \\ b_{m+1} \end{bmatrix} = \begin{bmatrix} 1 \\ a_m \\ 0 \end{bmatrix} + k_m \begin{bmatrix} 0 \\ b_m \end{bmatrix}
\]
\[
P_{m+1} = P_m + k_m \beta_m^*
\]

We talked about forward prediction:
\[
e_m^f[n] = x[n] + \sum_{k=0}^{\infty} a_k^{(m)*} x[n-k] = \sum_{k=0}^{\infty} a_k^{(m)*} x[n-k] (a_0^{(m)} = 0)
\]
\[
x[n] \xrightarrow{A(z)} e_m^f[n]
\]
\[
A(z) = \sum_{k=0}^{\infty} a_k^{(m)*} z^{-k} \text{ this is a non-phase filter}
\]
\[
\text{so it is invertible.}
\]

backward prediction:
\[
e_m^b[n] = x[n-m] + \sum_{k=0}^{\infty} b_k^{(m)*} x[n-k] = \sum_{k=0}^{\infty} b_k^{(m)*} x[n-k] (b_0^{(m)} = 1)
\]
\[
R_{m+1} \begin{bmatrix} a_m \\ 0 \\ b_m \end{bmatrix} = \begin{bmatrix} P_m \\ 0 \\ P_m \end{bmatrix} \]
\[
R_{m+1} \begin{bmatrix} \frac{P_m}{P_m^*} \\ 0 \\ \frac{P_m}{P_m^*} \end{bmatrix} = \begin{bmatrix} 0 \\ \frac{P_m}{P_m^*} \\ 0 \end{bmatrix}
\]
\[ p_m^a = p_m^b = p_m^c = a_m = J b_m^+ \]

**LDA (order-recursive procedure)**

\[
\begin{align*}
\alpha_x & \rightarrow \alpha_x \rightarrow \ldots \rightarrow \alpha_m \rightarrow \alpha_{m+1} \\
& \text{(want)}\ [p_m^a + h_m^+ b_m] = [p_m^a] + h_m^+ [p_m^a] = [p_m^a] + [0_{m+1}] \\
\bar{\beta}_m = \begin{bmatrix} r(1) & r(2) & \ldots & r(m) & 1 \\
\alpha_m \\
0 \\
\end{bmatrix}
\end{align*}
\]

So \( \beta_m + h_m^+ p_m = 0 \)

\[ K_m = -\frac{\beta_m}{p_m} \]

\[ \Rightarrow p_{m+1} = p_m + h_m^+ \beta_m^+ = p_m (1 - 1/K_m^2) \]

**So the procedure step:**

\[
\begin{align*}
\alpha_m & \rightarrow \beta_m, h_m = \frac{\beta_m}{p_m} \\
& \rightarrow p_{m+1} = p_m (1 - 1/K_m^2) \\
& [a_{m+1}^+] = [a_m^+] + h_m^+ [b_m^+] \\
\end{align*}
\]

Algorithm complexity:

- Compute \( \beta_m \): \( m \) mul + \( m \) add \( \mathcal{O}(m) \) every single step
- Compute \( h_m \): \( 1 \) division \( \mathcal{O}(1) \)
- Compute \( p_{m+1} \): \( 1 \) mul + \( 1 \) add \( \mathcal{O}(1) \)
- Compute \( a_{m+1} \): \( m \) mul + \( m \) add \( \mathcal{O}(m) \)

Total: \( m \) steps of \( \mathcal{O}(m) = \mathcal{O}(m^2) \)
notice $a_{m+1} = k_m$ (the last coefficient).

complexity per step: $(a_{m+1}) \text{mul}, (a_{m+1}) \text{add}, 1 \text{ div}

total comp: $\sum_{m=0}^{M-1} O(m) = O(M^3)$

example:

\[
R = \begin{bmatrix}
3 & 2 & 1 \\
2 & 3 & 2 \\
4 & 2 & 3
\end{bmatrix}
\]

\[
P_c = \ell(\ell) = 3
\]

\[
\begin{bmatrix} 1 \\ a_1 \end{bmatrix} = \begin{bmatrix} 1 \\ 0 \end{bmatrix} + k_0 \begin{bmatrix} 0 \\ 1 \end{bmatrix}, \quad \beta_0 = [\begin{bmatrix} \ell(1), \ell(0) \end{bmatrix}] = 2.
\]

\[
k_0 = -\frac{P_c}{P_1} = -\frac{3}{5} \quad \text{so:}
\]

\[
P_c = P_1 \left( 1 - \frac{3}{5} \right) = \frac{3}{5} = \frac{5}{3} \quad \left( \text{must be non-negative and decreasing} \right)
\]

\[
a_1 = \begin{bmatrix} \frac{2}{3} \end{bmatrix}
\]

\[
\begin{bmatrix} 1 \\ a_2 \end{bmatrix} = \begin{bmatrix} 1 \\ \frac{2}{3} \end{bmatrix} + k_1 \begin{bmatrix} 0 \\ -\frac{3}{5} \end{bmatrix}, \quad \beta_1 = [\begin{bmatrix} \ell(1), \ell(0), \ell(0) \end{bmatrix}] = 1 - 2 \cdot \frac{2}{3} = \frac{1}{3}
\]

\[
k_1 = -\frac{P_1}{P_2} = -\frac{1}{3} = \frac{1}{5}
\]

\[
P_2 = P_1 \left( 1 - k_1 \right) = \ldots
\]

\[
a_2 = \ldots
\]

Properties: for computing $\left( \begin{pmatrix} P_m \\ a_m \end{pmatrix} \right)$

1. We always have access to all predictors up to $M$

2. $P_{m+1} = P_m (1 - k_m P_m) \leq P_m \Rightarrow |k_m| \leq 1 \quad \forall m$

3. and if $P_{m+1} > 0 \Rightarrow |k_m| < 1$

$k_m$ - reflection coefficients (partial correlation coefficients)
3) $|k_m| = 1$, then $p_{m+1} = 0$ means:

$$R_{m+2} \begin{bmatrix} a_{m+1} \\ a_m \end{bmatrix} = \begin{bmatrix} a \\ a_m \end{bmatrix}$$

so $R_{m+2}$ singular - no longer positive definite.

This can continue with LDA (cannot divide by 0).

This is an efficient way to check positive definitiveness of a Toeplitz matrix (without calculating eigenvalues).

4) AR model of order $P$ then

$p_{p+l} = p_p$, $l \geq 1$ and

$k_p = k_{p+1} = \cdots = k_{p+l} = 0$.

and $A_{m+1}(z) = A_m(z)$

5) $r(x), r(n), \ldots, r(n)$ are equivalent representations of the same process. They contain same information.

If we want to encode/transform the process information, $z$ using $r(n)$ or $a_m$ is risky - the dynamic range is unknown, but $r(1)$ we know are in $[-1, 1]$ so we can quantize them well.

Going from repres to repres:

(1) $\rightarrow$ (2) $\rightarrow$ (3) by LDA.
3 \rightarrow 2 : \text{given } K_0 \text{ we can get from } P_m \text{ back to } P_0 = \frac{P_m}{1 - \frac{A}{K_m}}

Now, with } P_0, \text{ with } K_0 \rightarrow K_0 \rightarrow \ldots
\text{we can use the LDA to calculate } 3 \rightarrow 2

2 \rightarrow 1
extending the correlations beyond what is given is easily done by choosing the next } K's = K_m, K_{m-1}, \ldots
\text{if we choose zeros } K_0 = 0, l \geq M \text{ it means AR model assumption. This is also the maximum entropy extension.

}\begin{align*}
R_x(e^{i\omega}) &= A_m(e^{i\omega})^2 \\
\text{Reversal Levinson-Durbin:} \quad P_m &\rightarrow \begin{bmatrix} 1 \\ A_{m+1} \end{bmatrix} = \begin{bmatrix} A_m \\ 0 \end{bmatrix} + K_m \begin{bmatrix} b_m \\ 1 \end{bmatrix} \\
K_m &= A_m^{(m)} (K_m \text{ is the last entry of } A_m^{(m)})
\end{align*}
lets look at a pair of equations:

\[
a^{(m+1)}_2 = a^{(m)}_1 + k_m a^{(m)_*}
\]

\[
a^{(m+1)}_m = a^{(m)}_m + k_m a^{(m)_*} \rightarrow a^{(m+1)_*} = a^{(m)_*} + k_m a^{(m)}_2
\]

\[
\begin{bmatrix}
    a^{(m+1)}_1 \\
    a^{(m+1)_*}
\end{bmatrix} =
\begin{bmatrix}
    1 & k_m \\
    k_m^* & 1
\end{bmatrix}
\begin{bmatrix}
    a^{(m)}_1 \\
    a^{(m)_*}
\end{bmatrix}
\]

This way working on pairs of seq to solve for a

given a

Levinson Algorithm

we have \[x[n]\]

\[
x^{(m)}_n = \begin{bmatrix}
    x[n] \\
    x[n-m+1]
\end{bmatrix}
\]

\[
y[n] = c^* x^{(m)}_n
\]

\[
\text{still linear prediction}
\]

\[
R_{xx} C_0 = R_{xy} = \begin{bmatrix}
    R_{yy}[0] \\
    R_{xy}[0] \\
    \vdots \\
    R_{xy}[n-1]
\end{bmatrix} = E(x[n], y^{(m)}[n])
\]

Toepplitz

\[
R_m C_m = d_m
\]

\[
C_m \rightarrow C_{m+1}
\]

\[
c_m \text{ is easy (division) } x[0] C_0 = R_{xy}[0]
\]

know \[C_m\] we have solved \[R_m C_m = d_m\]

we want to solve \[R_{m+1} C_{m+1} = d_{m+1}\]

\[R_m \text{ is embeded in } R_{m+1}\]

\[
d_{m+1} = \begin{bmatrix}
    d_m \\
    d_{m+2}
\end{bmatrix} \text{ added 1 more entry .}
\]
Last class we solved find linear prediction using the Levinson-Durbin Algorithm (using the equations for feed forward pred). equivalence of parameter representation $x(0), x(1), \ldots, x(M-1) \Rightarrow r(0), a_1, a_2, \ldots, a_M \Rightarrow r(0), a_1, a_2, \ldots, a_M$ (better repr. when we need to quantize values).

Levinson Algorithm:

given $x(n)$, predict $y(n)$ (single scale).

$X =$ direct form FIR filter

$$X = \begin{bmatrix} x_n & x_{n-1} & \cdots & x_{n-M+1} \end{bmatrix}^T = X^T \begin{bmatrix} y_n \end{bmatrix}, \quad Y = y(n)$$

$y(n) = C_m^T X_m \begin{bmatrix} y(n) \end{bmatrix}$, the solution will be:

$$R_{x,x} \begin{bmatrix} x \end{bmatrix} = \begin{bmatrix} d \end{bmatrix}, \quad d_m = E(X_m y(n))^2$$
Rxx is Toeplitz, \( \mathbf{d_m} = \begin{bmatrix} r_{xx}(0) \\ r_{xx}(1) \\ \vdots \\ r_{xx}(s) \end{bmatrix} \)

We'll use an order-recursive procedure (like in LDA), recursive step:
assume we solved for \( \mathbf{C_m} \) \( (\mathbf{R_m} \mathbf{C_m} = \mathbf{d_m}) \)
and we want to find \( \mathbf{C_{m+1}} \) to solve \( \mathbf{R_{m+1}} \mathbf{C_{m+1}} = \mathbf{d_{m+1}} \)
\( \mathbf{R_{m+1}} \) has \( \mathbf{R_m} \) nested in it.
\( \mathbf{d_{m+1}} = \begin{bmatrix} \mathbf{d_m} \\ \mathbf{d_m} \end{bmatrix} \) just adding the \((m+1)^{\text{th}}\) scalar.

Try adding \( \mathbf{C_{m+1}} = 0 \):
\( \mathbf{R_{m+1}} \begin{bmatrix} \mathbf{C_m} \\ 0 \end{bmatrix} = \begin{bmatrix} \mathbf{d_m} \\ \mathbf{d_m} \end{bmatrix} \) we don't get \( \mathbf{d_m} \) as we want.
\( \psi_m = [r_{E_m}, \ldots, v_{E_m}, v_{E_m}] \begin{bmatrix} \mathbf{C_m} \\ 0 \end{bmatrix} = [r_{E_m}, \ldots, v_{E_m}] \mathbf{C_m} \)
we can calculate \( \psi_m \) easily.
if \( \psi_m = \mathbf{d_{m+1}} \), we're done. if not, we need some linear combinations.

satisfy:
\[ \mathbf{R_{m+1}} \begin{bmatrix} \mathbf{C_m} + k_m \mathbf{b_m} \\ 0 \end{bmatrix} = \begin{bmatrix} \mathbf{d_m} \\ \psi_m \end{bmatrix} + k_m \begin{bmatrix} \mathbf{0} \\ \mathbf{p_m} \end{bmatrix} = \begin{bmatrix} \mathbf{d_m} \\ \mathbf{d_{m+1}} \end{bmatrix} \]
we assume we already ran LDA, so we know \( \mathbf{b_m}, \mathbf{p_m} \)
\[ \psi_m + k_m \mathbf{p_m} = \mathbf{d_{m+1}} \rightarrow \]
\[ k_m = \frac{\mathbf{d_{m+1}} - \psi_m}{\mathbf{p_m}} \]
\[ \mathbf{C_{m+1}} = \begin{bmatrix} \mathbf{C_m} + k_m \mathbf{b_m} \\ 0 \end{bmatrix} \]
The complexity is also $O(m^2)$ (and since we need to run LDA $O(m^3)$ first, the total is still $O(m^3)$).

Lattice structures/filters:

$$e_{m+1}[n] = x[n] + \sum_{k=0}^{m^2} a_k x[n-k] = \left[ 1, a_{m+1} \right] x[n-1]$$

$$= \left[ 1, a_{m+1} \right] X_{m+1}(n) = \left[ 1, \left( \frac{a_m}{c} + k_m \left( b_m \right) \right) \right] x[n-1]$$

$$= \left[ 1, a_{m+1} \right] + k_m \left[ 0, b_m \right] X_{m+1}(n)$$

$$(1, a_{m+1}) X(n) + k_m \sum_{k=0}^{m^2} b_k x[n-k] =$$

$$= e_{m+1}[n] + k_m e_{m}[n-1]$$

$$e_{m+1}[n] = x[n-m-1] + \sum_{k=0}^{m^2} b_k x[n-k]$$

$$= x[n-m-1] + b_{m+1} x[n] = \left( b_{m+1} \right) X_{m+1}(n)$$

$$e_{m+1}[n] = \left( b_{m+1} \right) X_{m+1}(n)$$

So:

$$e_{m+1}[n] = e_{m}[n] + k_m e_{m}[n-1]$$

Similarly:

$$e_{m+1}[n] = e_{m}[n] + k_m e_{m}[n-1]$$
\[ e_0^f[n] \rightarrow e_1^f[n] \rightarrow \ldots \rightarrow e_M^f[n] \]
\[ e_0^b[n] \rightarrow e_1^b[n] \rightarrow \ldots \rightarrow e_M^b[n] \]
\[ x[n] \rightarrow e_1^b[n] \rightarrow e_1^b[n-1] \rightarrow \ldots \]
\[ e_{m-1}^f[n] \rightarrow e_m^f[n] \]
\[ e_m^b[n] \rightarrow e_m^b[n] \]

The relation is:
\[ A_m(z) = 1 + \sum_{k=2}^{M} a_k z^{-k} \]
\[ B_m(z) = \sum_{k=0}^{M} b_k z^{-k} \]
\[ e_m^b[n] = x[n-m] + \sum_{k=0}^{m} b_k x[n-k] \]

Transfer function between \( x[n] \) and \( e_m^f[n] \) is \( A_m(z) \)
Transfer function between \( x[n] \) and \( e_m^b[n] \) is \( B_m(z) \)

Book by P.P. Vaidyanathan - Multirate Systems & Filter Banks.
S.Mitra - DSP
lets take 2 -trans of eq (4.2).

- 2 -trans (4): $A_{m+2}(z)X(z) = A_n(z)X(z) + k_m^{-1}B_n(z)z^{-1}X(z)$
  getting $A_{m+1}(z) = A_n(z) + k_m^{-1}z^{-1}B_n(z)$ (4)

- 2 -trans (2): $B_{m+1}(z) = B_m(z)z^{-1} + k_mA_m(z)$

\[
\begin{align*}
A_{m+1}(z) &= \left[1, \frac{k_m}{z}, \left(\frac{A_n(z)}{z}\right)^{-1}\right] \quad (3+4) \\
B_{m+1}(z) &= \left[k_m, z^{-1}, B_n(z)\right]
\end{align*}
\]

**Properties:**
1. Modular structure (circuit implementation)
2. $|k_m| \leq 1$
3. All lower order predictors are available.
   (if you want to increase order - the previous $k$'s stay the same)
4. Can estimate $k$'s directly (no need to compute correlation)
   just solving minimization of $e_l^{m, n}$ or $e_l^{b, m}$ - select
   the minimizing $k_m$ value.
5. Orthogonality property of $e_l^{b, m}$
   $e_l^{b, m}\bot e_l^{b, n}$
   $e_l^{b, m}\bot e_l^{b, n}$, $l \neq m$

$E(e_l^{b, m}e_l^{b, n}) = \delta_l \delta_{l-m}$
(not true for fluid pred)
Proof: Assume $m > 0$.
\[ e_m = x_m - x_{m-1} + b_m X_m \]
\[ e_m = \begin{bmatrix} e_m \end{bmatrix}_m \]
\[ \mathbf{e}^T = [x_0 \ldots x_m] + [x_0 \ldots x_m]^T \]
\[ X_m = \begin{bmatrix} X_{m+1} \mid \text{extra} \end{bmatrix} \]

(c) Cholesky Factorization of $R_m$
\[ e_m = x_m \]
\[ \begin{bmatrix} e_m \mid e_{m+1} \mid \vdots \mid e_{m+n} \end{bmatrix} = \begin{bmatrix} \mathbf{I}_m & & & & & & \frac{1}{2} \mathbf{I}_m \end{bmatrix} \begin{bmatrix} X_m \mid X_{m+1} \mid \vdots \mid X_{m+n} \end{bmatrix} \]

Lower triangular matrix calculated from LDA coeff.
\[ e_m = L X_{m+1} \]

a. Linear mapping between original data $x_m$ and the backward pred. errors.

b. $L$ is an invertible matrix.

You can store the errors $e$ and reconstruct $x$'s from them.

correlations of this vector equation:

\[ \begin{bmatrix} P_0 & \ldots & 0 \\ 0 & \ddots & \vdots \\ 0 & \ldots & P_n \end{bmatrix} = L R_{m+1} L^H \]
\[ D = LR_{m+1} L^H \]

So:
\[ R_{m+1} = L^{-1} D L \]

A way to invert a PD Toeplitz matrix.

\[ R_{m+1}^{-1} = L^H D^H L \]
add 17.14 to HW3.
Next Tuesday-Midterm exam.

Levinson Algorithm

Estimate \( y_{\text{hat}} \) given \( x[0], \ldots, x[n-\alpha+1], x[n] \)

\[
\hat{y}[n] = C_m^T x_m[n]
\]

we had efficient algo to solve \( C_m \)

\[
x(n) \xrightarrow{H(z)} \hat{y}[n] = \sum_{k=0}^{M-1} h(k) x(n-k)
\]

\[
h[k] \leftarrow C_m \begin{bmatrix} h[0] \\ h[1] \\ \vdots \\ h[\alpha-1] \end{bmatrix} = C_m^T
\]

so we can think of this as input time series \( x[n] \) and filter to get output seg \( y[n] \).

solution \( R_{xx} C_m = R_{xy} \)

Estimate \( y[n+\alpha] \)

\begin{align*}
\alpha > 0 & \quad \text{prediction} \\
\alpha = 0 & \quad \text{filtering} \\
\alpha < 0 & \quad \text{smoothing}
\end{align*}

\( M = \infty \) \quad \text{causal Wiener Filter}

we can also discuss non-causal Wiener filter

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

not implementable, but can give us theoretical bounds- how well can we possibly do.
\[ E(\|\hat{Y}(n)\|^2) = E(\|Y(n)\|^2) - R_{yx} R_{xx}^{-1} R_{xy} \]

If \( x, y \) are uncorrelated \( R_{xy} = 0 \), \( c \to 0 \) \( \to \) no prediction at all \( \to \) worse performance.

* Notice: the solution of a single coefficient filter \( c \) assumes \( x, y \) are jointly ws's, otherwise \( c \) would have to be a func of time \( \leq m \).

Practically not used practically assume joint ws's. Unless modeling time varying systems.

\[
\hat{y}(n) = y(n) - \hat{y}(n) = x(n) + m
\]

orthogonality principle

\[
E(\hat{y}(n) x^*(m)) = 0 \Rightarrow E(\{y(n) - \sum_{k=-\infty}^{\infty} h(k) x(n-k)\} x^*(m)) = 0
\]

\[
E(\hat{y}(n) x^*(m-n)) = 0 \Rightarrow E(\{y(n) - \sum_{k=-\infty}^{\infty} h(k) x(n-k)\} x^*(n-m)) = 0
\]

\[
R_{yx}(e^{jw}) = H_0(e^{jw}) R_{xx}(e^{jw})
\]

\[
H_0(e^{jw}) = \frac{R_{yx}(e^{jw})}{R_{xx}(e^{jw})} \quad \text{(non causal Wiener filter)}
\]

\[
E(\|\hat{y}(n)\|^2) = E(\hat{y}(n)(\hat{y}(n) - \hat{y}^*(n)) = E(\hat{y}(n) \hat{y}^*(n)) - 0 = \int_{-\pi}^{\pi} R_y(e^{jw})(1 - |G_{yx}(e^{jw})|^2) dw
\]
\[ (g_x(e^{i\omega}))^2 = \frac{|R_{yx}(e^{i\omega})|^2}{R_{xx}(e^{i\omega})R_{yy}(e^{i\omega})} \quad \text{Coherence function} \]

The coherence function tells us how much predictability we possibly have, for every frequency \( e^{i\omega} \).

back to Levinson algò:
\[ R_{m+1} C_{m+1} = d_{m+1} \quad \text{recursion:} \]
\[ C_{m+1} = \begin{bmatrix} C_m \\ 0 \end{bmatrix} + k_m \begin{bmatrix} b_m \\ 1 \end{bmatrix} \]

Lattice filters:
\[ C_{m+1}[n] = x[n] + \sum_{k=1}^{m+1} a_k \cdot x[n-k] = \sum_{k=0}^{m+1} a_k \cdot x[n-k], (a_{m+1} = 0) \]
\[ C_{m+1}[n] = x[n] - \sum_{k=0}^{m+1} b_k \cdot x[n-k], (b_{m+1} = 1) \]

this is the direct form filter for the final and backward prediction error.

If I wanted to increase order by 1, I'd have to recompute all the coefficients.

This structure lacks the recursive form of the LDA.
The lattice way:

\[ e_{m+1}^f(n) = e_m^f(n) + k_m x e_m^b [n-1] \]

\[ e_{m+1}^b(n) = e_m^b(n-1) + k_m e_m^f(n) \]

This is a modular structure in each module (block):

\[ e_{m+1}^f(n) \quad e_{m+1}^b(n) \]

Good properties in the lattice form:

The big solution (large order) contains the values of the lower order filters embedded inside.

\[ k_m \text{ values in } E_{1,1} \Rightarrow \text{nice to quantize.} \]

\[
\begin{align*}
\{ e_{m+1}^f(n) &= (1, b, m^t) X_{m+2}(n) \\
 e_{m+1}^b(n) &= (b^t, 1) X_{m+2}(n) \\
\}
\end{align*}
\]

We'll see some more good properties:

\[ e_m^b(n) \perp e_l^b(n) \quad l \neq m \]

\[ E(e_m^b(n) e_l^b(n)) = P_m \delta_{m-l} \]

\[
\begin{bmatrix}
 e_0^b(n) \\
 \vdots \\
 e_m^b(n) \\
\end{bmatrix} = L \begin{bmatrix}
 X(n) \\
 \vdots \\
 X(n-m) \\
\end{bmatrix} \quad e_{m+1}^b(n) = L X_{m+1}(n)
\]

lower triangular matrix.
$
L = \begin{pmatrix} 1 & 0 \\ b & 0 \end{pmatrix} \rightarrow \text{we can compute } L \text{ by}

\text{the } b \text{'s we calculate in LDA.}

\text{correlated}

X_{[b]} \text{ } \rightarrow \text{ } \frac{X_{[b]} - X_{[b]}}{1}

E_{[b]}^*, E_{[b]}^* = X_{[b]}^* - X_{[b]}^* (\text{only what is new})

\text{uncorrelated}

\rightarrow \quad X_{[b]}^* - X_{[b]}^*

\text{new info}

\rightarrow \quad X_{[b]}^* - X_{[b]}^*

\text{there's a map between } X \text{'s and } e^*_b \text{'s.}

\text{Estimating } y \text{ as given } X_{[b]} \text{ is identical to}
\text{estimating } y \text{ as given } E_{[b]}^* \text{.}

\text{Linear prediction using invertible linear transformation of data:}

\rightarrow \quad X \rightarrow Z = M X \quad (M \text{ invertible})

\rightarrow \quad \Phi = C_{x}^* X \quad \rightleftharpoons \quad \Psi = C_{z}^* Z

\rightarrow \quad R_{xx} C_{x} = R_{xy} \quad \rightarrow \quad R_{zz} C_{z} = R_{zy}

\text{These will produce the same estimate } \hat{y} \text{ with same quality of estimation.}

\text{The difference can be in computation resources.}

\text{When transforming can show: } C_{x}^* X = C_{z}^* Z = C_{z}^* M X

\text{or: } \quad C_{x}^* = C_{z}^* M \cdot
orthogonality of backward pred error

\[ E(e_{m+1}^b[n] e_{m+3}^b[n]) = L R_{m+2} L^H = \begin{bmatrix} P_0 P_1 & 0 \\ 0 & P_m \end{bmatrix} \]

\[ \det R_{m+1} = \prod_{i=1}^m P_i \]

\[ \frac{\det R_{m+1}}{\det R_m} = P_m \]

We prefer estimating \( y_{m+2} \) using \( e_{m+3}^b[n] \), (calculated by the LDA):

\[ R_m e_{m+3}^b \cdot h_m = R_m e_{m+3} y_{m+3} \]

\[ \begin{bmatrix} P_{m+2} & 0 \\ 0 & P_{m-1} \end{bmatrix} \begin{bmatrix} h_{m+2} \\ \vdots \end{bmatrix} = \begin{bmatrix} E(e_{m+3} y_{m+3}) \\ \vdots \end{bmatrix} \]

Diagonal corr matrix.

the solution is easy to compute:

\[ h_m = \frac{E(e_{m+3} y_{m+3}^*)}{P_m} \]

\[ E(e_{m+3} y_{m+3}^*) = E(e_{m+3} y_{m+3}) \]

If we construct the lattice filter we have access to all the \( e_0 \) values.

\[ y_m[n] = \sum_{k=0}^{m-1} h_k^* e_{m-k}[n] \]

\[ y_m[n] = \sum_{k=0}^{m-1} h_k^* e_{m-k}[n] = y_{m-1}[n] + h_m^* e_{m-1}[n] \]

order-recursive: if we already calculated estimate using \( m \) coeffs, we can improve by just adding the new information.
\[ y_{m+1}[n] = c_{m+1}^H x_{m+1}[n] = [c_{m+1}^H, 0] x_{m+1}[n] + k_m^c [b_m^x][x_{m+1}[n]] = \]
\[ = y_m[n] + k_m^c e_m[n] \]

(old estimate, based on m samples) (the correction based on 1 extra value)

So \[ k_m^c = h_m^c \] \[ K_m^c = h_m \]
so now we have an interpretation to these correction coefficients \( K_m^c \).

"Lattice filter: good."

\[ c_{m+1}[n] = e_m[n] + k_m^c e_m[n-1] \]

\[ e_{m+1}[n] = e_m[n-1] + k_m e_m[n] \]

right now we know to: data \( \rightarrow \) estimate correlations \( \rightarrow \) levinson durbin \( \rightarrow \) \( h_m \)’s

now we’ll see how to get the \( h_m \)’s from data\( \sqrt{} \).

Transfer functions: \[ A_{m+1}(z) \] (we know this is min-phase)

\[ B_{m+1}(z) = z^{-m+1} A_{m+1}(\frac{1}{z})^* \] (since \( a_{m+1} = J b_{m+1}^* \))
A(\xi) is min-phase
B(\xi) is max-phase

(only other \xi)

Can estimate \gamma(\omega) using e_0^b(\omega_1), \ldots, e^b(\omega_n) instead of using x[n], x[n-1], \ldots, x[n-w]

\gamma(\omega) = \sum_{k=0}^{w} h_k e_k(\omega_n)

he are decoupled (can solve for each l independently)

\gamma = \frac{E(e^b(x) y^*)}{p_l}

data X \rightarrow transformed data Z = M X \quad M invertible

then \quad \gamma = C_x^H X = C_x^H \mathbf{Z} = C_x^H M X

we just have to decide which (C_x or C_z) is easier to compute.

C_x^H = C_x^H M \quad or \quad C_x = M^H C_z

let's check:

M^H C_z = M^H R_z R_z^{-1} R_z y = M^H (R_x M_x M^H)^{-1} R_x M_x y = M^H M_x R_x M^H M_x^{-1} M_x R_x y

= R_x^{-1} R_x y = C_x^H

Estimate k_m:

\frac{1}{e_m^n(x)} \frac{1}{e_m^n(z)} \quad \text{linear}(x[n-1], x[n-2], \ldots, x[n-w])

orthog to linear comb of x[n-1], x[n-2], \ldots, x[n-w]

E(e^f(x) e^b(x) e^b(x))^* = 0

o = E(e^f(x) e^b(x) e^b(x))^* + k_m E(1 e_m^n(x))^2

so

k_m = \frac{E(e^f(x) e^b(x) e^b(x))^*)}{E(1 e_m^n(x))^2}

\boxed{q}
Let's see if this formula is useful:

\[ K_m = -\frac{\sum x e_m^{a \cdot} b e_m^{b \cdot} (n-1)}{\sum e_m^{b \cdot} (n-1) \cdot} \]

Start with \( K_0 \):

\[ K_0 = -\frac{\sum x e_m^{a \cdot} (n-1)}{\sum (x e_m^{a \cdot})^2} \]

Then we compute \( e_m^{a \cdot}, e_m^{b \cdot} \) using the lattice filter module and then use the formula for \( k_4 \).

We can show, again (\( c(x) \)): \( e_m^{b \cdot}(n-1) \cdot e_m^{f \cdot}(n) \)

We can use this to get a new estimation of \( k \): \( c(x) \):

\[ c = E(e_m^{b \cdot} - e_m^{f \cdot}) + k_m E(1 e_m^{b \cdot}(n) \cdot) \]

\[ K_m = -\frac{E(e_m^{b \cdot} - e_m^{f \cdot} e_m^{a \cdot} (n-1))}{E(1 e_m^{f \cdot}(n) \cdot)} \]

The numerator is the same in both formulas, the denominator is different. They are equal theoretically, but may be different when estimating from finite sample:

\[ K_m = -\frac{\sum e_m^{b \cdot} - e_m^{f \cdot} e_m^{a \cdot} (n-1)}{\sum e_m^{f \cdot}(n) \cdot} \]

From LDA:

\[ (1) - \frac{B_m}{P_m} \]
now we have 2 ways to estimate $k^*$ from finite data. We can also take some combinations.

Other estimates:

We want to ensure the estimate will result in $k_s$ that are bounded by $|k_s| \leq 1$.

$\tilde{k}_m = \frac{E(\eta_k \tilde{\eta}_k^* \tilde{\eta}_k^{\tilde{\eta}_k})}{\sqrt{E(\eta_k \tilde{\eta}_k^*)^2 E(\tilde{\eta}_k \tilde{\eta}_k - 1)^2}}$

use geometric mean of two estimates of $\eta_k$

this $\tilde{k}_m$ is bounded by $1$ (Cauchy-Schwarz)

$\tilde{k}_m = \frac{E(\eta_k \tilde{\eta}_k^*) + E(\tilde{\eta}_k \tilde{\eta}_k - 1)^2}{2}$

use arithmetic mean.

this $\tilde{k}_m$ is also bounded by $1$ (Burg estimate)

now we can: $\tilde{S} \rightarrow k_s \rightarrow a_m \rightarrow \text{Power spectrum}$

We can pose this approach as an optimization problem

$J(k_m) = \alpha E(\eta_k \tilde{\eta}_k^*) + (1-\alpha) E(\tilde{\eta}_k \tilde{\eta}_k - 1)^2$

$\alpha \in [0, 1]$ can result in $a = 1, 2, 3$.

different choices of $\alpha$ will result in $a = 1, 2, 3$. 
Midterm:

- 1. A sheet of notes (no solutions to problems)
- Material: anything before lattice filters, covered by homeworks
  - broadly: 1. Modeling (AR, MA, ARMA)
  - 2. Linear Mean Square estimation
  - 3. Application to time series - fundamental pred.
    - LDA, Levinson Alg.

---

Let's go over some HW prob.

\[ G(z) = \frac{1 - 0.8 z^{-1}}{1 - 0.9 z^{-1}} \]

\[ \hat{y}(m) = \frac{1}{\sigma_n^2} y(m) \]

\[ y(m) \rightarrow x(m) \]

Interested in \( s(m) \) (our sensor) (and) (also noise)

we have \( x(m) \), we wish to estimate \( s(m) \).

book says 2nd-order filter:

\[ x(m) \rightarrow \left( c_0 + c_1 z^{-1} \right) \]

\[ s(m) \]

\[ X = \begin{bmatrix} x[m] \\ x[m-1] \end{bmatrix}, \quad Y = [s] \]

solution: \( R_{xx} \leq R_{xy} \)

\[ R_{xx} = \begin{bmatrix} r_{x} & r_{x}[1] \\ r_{x}[1] & r_{x}[2] \end{bmatrix}, \quad R_{xy} = \begin{bmatrix} r_{xs}[0] \\ r_{xs}[1] \end{bmatrix} \]

\[ x[m] = y[m] + v[m] \rightarrow R_{xx}[m] = r_{y}[m] + r_{v}[m] = r_{yy}[m] + \sigma_v^2 \]

\[ s[m] = x[m] + u[m] \rightarrow r_{xs}[m] = r_{ys}[m] \]

\[ s[m] = x[m] + u[m] \rightarrow r_{xs}[m] = r_{ys}[m] \]
\[ w(n) = \sqrt{1 - 0.82^{-4} - 0.92^{-2}} \]

\[ R_{yy}(z) = \sum_{n=0}^{\infty} (1 - 0.82^{-4})(1 - 0.92^{-2})(1 - 0.82)(1 - 0.92) z^{-n} \]

\[ R_{yy}(z) = 0 \quad \text{for} \quad |z| > 2, \quad \therefore \]

we can also compute in the time domain (difference equation)

* Orthogonality principle

in class we said: \( y_o = c^H x \) when \( R_{xx} C = R_{xy} \)

\[ y_o = y - y_o \] and claimed \( y_o = y^H B x \)

in hw we do reverse: we start with orthogonality and show that the \( C \) has to satisfy \( R_{xx} C = R_{xy} \)

given: \( C \) satisfies \( \tilde{y} = y - C^H B x \) (not \( y \))

show that \( C \) satifies \( R_{xx} C = R_{xy} \)

we know \( E(\tilde{y} B^H) = 0 \) \[ E((y - C^H x) B^H) = 0 \] \[ E((y - C^H x) x^H) B^H = 0 \]

so \[ E((y - C^H x) x^H) = 0 \rightarrow R_{xx} - C^H R_{xx} = 0 \rightarrow R_{xx} C = R_{xy} \]

\[ R_{xx} C = R_{xy} \]

* General vector case: \( y \) is vector orthog: \( E(x y^H) = 0 \rightarrow \forall \delta, x \perp y \)

---

**Non-Stationary Signal**

1) Slowly Varying (LMS algo)
2) Modeling & then optimal estimation (Kalman Filter)

\[ x[n] \xrightarrow{h[n]} o \xrightarrow{} y[n] \] (Equalization)

Changing
we would like to estimate/reconstruct $x(n)$ from $y(n)$ (equalization) $\rightarrow \hat{y}(n) \rightarrow \{c(n)\} \rightarrow x(n)$

however now we talk about a system/Filter $h(n)$ which is time-varying, so we also have to adapt the equalizer $c(n)$. ($y(n)$ is not wss)

Narrowband signals in wide band data:

$X(n) = S(n) + V(n)$

narrowband signal with unknown freq and num of components
data (wide band) like white noise

so we may be interested in the data $V(n)$ which is interfered by the sinusoids in $S(n)$.

the basic assumption: $S(n)$ is predictable while $V(n)$ is not so match

$\hat{X}(n)$ should be a filter to predict $S(n)$ from past samples of $x(n)$.

$H(z)$ may have to change through time if $S(n)$ is not wss.
Some New Issues
(Said's lectures on adaptive filters)
and book ch. 9.

1. $c(n)$ is not constant. $c(n)$ \rightarrow asymptotic behavior
   \[ E(c(n)) \xrightarrow{n \to \infty} \]
   \[ \text{Cov}(c(n)) \xrightarrow{n \to \infty} \]

2. Tracking Ability - Excess Mean Squared Error.
3. Computation - range of solutions.

Problem: Predict $y(n)$ given $x(n), \ldots, x(n-M+1)$
\[ P(c) = E(|y(n) - c^H x(n-M+1)|^2) = y(n, n) - c^H R_{xy}[n] - R_{xx}[n] c + c^H R_{xx}[M] c \]
(These are functions of $n$, not constants). When we solve this (minimize MSE) we get:
\[ R_{xx}[n].c_{0} = R_{xy}[n] \]
now $c_{0}$ is a function of time: $c_{0}(n)$

Challenge
4. Compute $R_{xx}[n] \land R_{xy}[n]$ (either from given model
   or estimate from our very limited data)
   possible: compute over segments of data.

2. $R_{xx}[n]$ doesn't have to be Toeplitz anymore.
3. Take advantage of previous estimates (e.g., from previous data) Segments
(fix/refine)

Calculate it based on C_{(n)} tracking.

Potential solution:
1. Assume stationary over time segments.
2. How to select segments?
   a. Non-overlapping
   b. Overlapping

(P.E.\text{\textsc{in}})

P.E.\text{\textsc{is}} quadratic in c. \( E \{y(t)|c^n x_m, i_1^n \}^2 \)

(\text{P.E.\text{\textsc{in}}})

\text{Gradient descent approach.}

take the solution from previous time and compute the gradient in the new time, to do the small adjustment.

We need the concept of derivatives for complex coefficients.

(appendix of book)

(Stochastic Approximation Algorithms)
derivative for real vectors

\[ P(c) \text{ when } c \in \mathbb{R}^n \]

gradient:
\[
\nabla_c P(c) = \left( \frac{\partial P}{\partial c_1}, \ldots, \frac{\partial P}{\partial c_n} \right)
\]

eg:
\[
\nabla_c (x^T A c) = A x
\]
\[
\nabla_c (c^T A y) = A y \quad \text{if } A \text{ is symmetric}
\]
\[
\nabla_c (c^T A c) = A^T c + A c = 2A c.
\]

What if \( c \) is complex \( c \in \mathbb{C}^n \)

is \( P(c) \) differentiable?

presence of \( \bar{c} \) makes the function not differentiable.

Need Cauchy-Riemann conditions

first consider \( c \) being a complex scalar

\[ P(c) = c^* = a - j b \quad (c = a + j b) \]

\[ P(c) = u(x, y) + j v(x, y) \]

\[
\begin{bmatrix}
\frac{\partial u}{\partial x} & \frac{\partial v}{\partial y} \\
\frac{\partial u}{\partial y} & -\frac{\partial v}{\partial x}
\end{bmatrix}
\]

this is the condition for differentiability

in our case \( u = a, v = -b \), we get:

\[
\frac{\partial u}{\partial x} = 1, \quad \frac{\partial v}{\partial y} = -1 \quad \frac{\partial u}{\partial y} = 0, \quad \frac{\partial v}{\partial x} = 0
\]

\[
\frac{\partial u}{\partial c} = 1 \neq \frac{\partial v}{\partial c} = 1
\]

so the condition doesn't hold \( \Rightarrow P(c) \) is not differentiable.
we can also use the definition of the derivative:

$$\lim_{\Delta c \to 0} \frac{P(c + \Delta c) - P(c)}{\Delta c} = \lim_{\Delta c \to 0} \frac{(a + \Delta a) - j(b + \Delta b) - (a - j b)}{\Delta a + j\Delta b}$$

$$= \lim_{\Delta c \to 0} \frac{\Delta a - j \Delta b}{\Delta a + j \Delta b}$$

this limit depends on the path of $\Delta c \to 0$. It can be $-1$, $e^{i\pi}$, in different paths.

no unique limit $\Rightarrow$ there is no derivative.

We do not need the traditional derivative. We can define a new derivative, which is more convenient: Wirtinger Calculus.

start with stationary case:

$$P(c) = E(\gamma y^2) - \gamma^2 R_{yy} R_{xx} C + \gamma^2 R_{xx} C$$

real valued function.

$c = a + jb$, then we can re-describe $P(c)$ as:

$P_0(P(a, b)) = \text{real valued function in } a, b$ (both real arguments)

so we can apply calculus of real variables to $P_0(a, b)$

we can solve for $a, b$, and then $c_0 = a + jb$.

**define new derivatives:**

$$\frac{\partial P}{\partial c} = \frac{1}{2} \left[ \frac{\partial P}{\partial a} - j \frac{\partial P}{\partial b} \right]$$

$$\frac{\partial P}{\partial c^*} = \frac{1}{2} \left[ \frac{\partial P}{\partial a} + j \frac{\partial P}{\partial b} \right]$$
Let's try using these derivatives:

\[ \nabla_c (P_c) = \frac{2P}{3c} \]

(when \( P \) changes)

\[ \nabla_{c^*} (P_c) = \frac{2P}{3c^*} \]

\[ \nabla_c (b^* c) = b^* \]

\[ \nabla_c (y^* c) = y^* \]

\[ \nabla_{c^*} (c^* R_c) = R_c \]

So we can think of \( c \) and \( c^* \) as two separate variables.

Back to \( P(c) \).

\[ \nabla_{c^*} P(c) = -R_{xy} + R_{xx} c = 0 \]

We get the same equation that we know from real values.

---

Gradient descent

\[ \frac{\partial P(c)}{\partial c} \]

\[ P(x) \text{ real func of } x \]

\[ P(x_2) = P(x_1) + (x_2 - x_1) \frac{\partial P}{\partial x} \bigg|_{x=x_1} + \frac{1}{2} (x_2 - x_1)^2 P_{xx}(x_2 - x_1) \]
going over midterm

3) \( X[n] = X[n-1] - 0.5 \cdot X[n-2] + w[n] \)
\( X^2[n] = -a_m \cdot X[n-1] \) \( m = 5 \)
the best predictor for the AR model is 5
so: \( \hat{P}_2 = \hat{P}_4 = \hat{P}_5 \)

\( P_m = P_{m-1} \cdot (1 - |k_m|) \)
so \( k_2 = k_3 = k_4 = 0 \)

\( a_m = \begin{bmatrix} -1 \\ 0.5 \end{bmatrix} \) so \( k_1 = 0.5 \)

now \( \begin{bmatrix} 0 \\ 0 \end{bmatrix} = \begin{bmatrix} 0 & 0 \\ 0 & 0 \end{bmatrix} \) \( \begin{bmatrix} a_1 \\ 0 \end{bmatrix} + k_1 \begin{bmatrix} a_1 \\ 0 \end{bmatrix} \) getting \( a_1 \) and \( k = 0 \)

2)b. \( \hat{X}[n] = -a_2 \cdot X_2[n-1] \) we should use LDA

to find \( a_2 \)

2)c. \( W[n] \xrightarrow{H(z)} \) the prediction is whitening of \( X[n] \)

\( A_m(\hat{z}) \) should invert \( H(z) \) (FIR) so \( A_m(\hat{z}) \) has to be IIR - need more to improve.

max: 127, min: 18, mean: 30, std: 7.7

Back to adaptive filters.
start with MSE:

\[ P(\varepsilon) = E[|Y[n]|^2 - C^H \cdot X \cdot c_c]^2 = E(Y[n])^2 - C^H \cdot R_{xy} \cdot C_c \]
\[ = E(Y[n]) - R_{xy} \cdot R_{xx}^{-1} \cdot R_{xy} + (C_c)^H \cdot R_{xx} \cdot (C_c - C_0) \]
\[ (R_{xx} - C_0 = R_{xx}) \]
\[ P_0 = \text{optimum} + \hat{C}_c^H \cdot R_{xx} \cdot \hat{C}_c \]
\[ P(\varepsilon) = P_0 + \hat{C}_c^H \cdot R_{xx} \cdot \hat{C}_c \]
we'll want $c$ to converge to $c_0$.

non-stationary:

$$\mathbf{P}(t,n) = E[|y(n)|^2] - R_{yy}(n) - R_{xx}(n) + (c - c_0)^H R_{xx}(c - c_0)$$

(when $R_{xx}(c_0) c_0 = R_{yy}(c_0)$)

$$R_{yy}(n) = E\left( x^H(n) y(n) \right)$$

$$R_{xx}(n) = E\left( x^H(n) x(n) \right)$$

everything is now a function of time $n$.

the optimum solution is time-varying.

- Assumption: $c_0(n)$ is slowly varying
  (i.e., $P(c_0(n), n)$ is also slowly changing).
- Gradient-based procedure provides a convenient update procedure for $c_0(n)$.

Steepest Descent Methods:

- based on Taylor series

$$p(c) = \frac{\partial f}{\partial c}$$

say we want to minimize $f(x)$, we have

a guess $x_1$: $f(x_1) \approx f(x_1) + \frac{\partial f}{\partial x_1}x_1$ $f(x_1)$

valid in close neighborhood of $x_1$. 
we want to reduce \( (x_2 - x_0) \frac{\partial f}{\partial x} \bigg|_{x=x_2} \) to be negative and \( \theta \) as far from 0 as possible. The best guess is \( (x_2 - x_0) \) will be exactly negative to the gradient \( \frac{\partial f}{\partial x} \bigg|_{x=x_2} \) (times some factor).

\[ x_2 = x_1 - \mu \frac{\partial f}{\partial x} \bigg|_{x=x_2} \]

\[ \begin{align*}
\frac{2(p(x))}{2c^{*}} &= \frac{1}{2} \left[ \frac{2p(c)}{2a} + 1 \right] \\
\frac{2(p(c))}{2b} &= -R_{xy} + R_{xx} c
\end{align*} \]

\[ c = a + ib \]

the gradient update will be:

\[ C_k = C_{k-1} - \mu \left[ R_{xy} + R_{xx} C_{k-1} \right] \quad (\mu > 0) \]

for non-ws's:

\[ C_k = C_{k-1} - \mu \left[ -R_{xy} C_{k-1} + R_{xx} C_{k-1} \right] \]

\[ \text{(wss):} \]

\[ C_{k-1} = E(x_m(k) x_{m+1}(k)) \] \( C_k \) = \( E(x_m(k) y_{m+1}(k)) = E((x_m(k))^T (x_{m+1}(k)) \)

\[ = E(x_m(k) e^{*}(k)) \]

\[ (e(k)) = y(k) - C_{k-1} x_{m}(k) \)

\[ S_0 = \mu E(x_m(k) e^{*}(k)) \]

\[ C_k = C_{k-1} + \mu E(x_m(k) e^{*}(k)) \]
now do the same for non-w's:

\[ C_k = C_{k-1} + \mu E[R_{xy}(k) + R_{xx}(k) C_{k-1}] = \]

\[ = C_{k-1} + \mu E[X_{11}(k)e(k)] \]

we get the same result for the update, only now the expectation varies with time.

This is a stochastic gradient descent algorithm (SDA).

Least Mean Square Algorithm: (LMS)

\[ C[n] = C[n-1] + \mu X_{n} e_n^{*} e_n \] when \( e_{n} = y_{n} - c^{*} x_{n} \)

Simply we got rid of the expected value and now using the instantaneous (random) values of \( X_{n} e_n \) and \( e_n \).

so now \( C[n] \) is a random sequence (the \( c_n \) from before was deterministic).

Understand the LMS:

1) convergence \( C[n] \overset{\text{prob}}{\rightarrow} \) (any options \( C[n] \overset{\text{in mean}}{\rightarrow} \) \( C[n] \overset{\text{in mean-squared}}{\rightarrow} \))

2) selecting \( \mu \)

First well study SDA (deterministic case) and only then the LMS's.
notice a challenge for LM's: needs the true y(k) value (in order to update to C(k+1)) whether in the SPA using only statistics of x and y.

Clever engineering: generate a good reference algorithm.

(Adaptive noise cancellation)

\[ \begin{align*}
S(h) &\xrightarrow{H(z)} v(n) \\
&\xrightarrow{0} x^n \\
&\xrightarrow{C(z)} s[n] \in \mathbb{R} \\
&\xrightarrow{+\hat{y}(n)} y[n]
\end{align*} \]

\textbf{w.c.'s case:}

\[ C_k = C_{k-1} + \mu \left( R_{xy} - R_{xx} C_{k-1} \right) \quad \lim_{k \to \infty} C_k = ? \]

in what conditions \( C_k \to C_0 = R_{xx}^{-1} R_{xy} \)?

and how to select \( \mu \)?

\( \mu \) large \( \rightarrow \) speed up the update, but the approximation is less accurate. tradeoff.

\[ C_k - C_0 = C_k \xrightarrow{?} 0 \quad \hat{e}_k = \hat{e}_{k-1} + \mu \left( R_{xy} - R_{xx} C_{k-1} \right) \]

\[ \hat{e}_k = \hat{e}_{k-1} + \mu \left( R_{xy} C_0 - R_{xx} C_{k-1} \right) = \hat{e}_{k-1} - \mu R_{xx} \hat{e}_{k-1} = (I - \mu R_{xx}) \hat{e}_{k-1} \]

this tells us how the errors are propagating in time.

\[ \hat{e}_k = (I - \mu R) \hat{e}_{k-1} \]

for convergence we need \( |\lambda(I - \mu R)| < 1 \) \( \Rightarrow \)

\[ R = Q \Lambda Q^H, \quad \hat{e}_k = Q(I - \mu \Lambda) Q^H \hat{e}_{k-1} \]

so:

\[ QQ^H = \begin{pmatrix} x^* x & 0 \\ 0 & 0 \end{pmatrix} \]

\[ \begin{pmatrix} \Lambda & 0 \\ 0 & \Omega \end{pmatrix} \] \( Q^H \hat{e}_k = (I - \mu \Lambda) Q^H \hat{e}_{k-1} \]

\[ \hat{e}_k = (I - \mu \Lambda) \hat{e}_{k-1} \]
Last time we examined steepest descent algo., for \textit{wss}:

\[ P(c) = E \left( \frac{1}{y(n)} - x(n) \varepsilon(n)^2 \right) = E \left( \frac{1}{y(n)} \right) - R_{xy} c - c^* R_{yy} c = P_0 + \varepsilon^* R_{xx} \varepsilon \]  
\[ (\varepsilon = c - c_0) \]

\[ \nabla c^* P(c) = R_{xx} c - R_{xy} = -E(x(n) \varepsilon(n)^* \varepsilon) \]

so the gradient descent (steepest descent) update is:

\[ c_k = c_{k-1} - \gamma \mu \nabla c^* P(c) | c = c_{k-1} \quad (\mu > 0) \]

(now he changed to "\( \gamma \mu \)" to be consistent with the book's version),

so:

\[ c_k = c_{k-1} - \beta \mu (R_{xx} c_{k-1} - R_{xy}) \]

the radical version to the SDA was the \textit{LMS} algo.; it is where we don't take any expectation, but just use the instantaneous values.

\textit{LMS}:

\[ c(n) = c(n-1) + \beta / x(n) e^*(n) \]

\[ (e(n) = y(n) - c^*(n-1) x(n)) \]

\textit{LMS} is a stochastic version: each time step it updates using the random realization of the next sample.
side comment \( x[n] = \begin{bmatrix} x[n] \\ x[n-1] \\ \vdots \\ x[n-M] \end{bmatrix} \)

a more general version:
\( x'[n] = \begin{bmatrix} x_1[n] \\ x_2[n] \\ \vdots \\ x_M[n] \end{bmatrix} \)

it's a different problem now.

- A popular variant: Normalized LMS (NLMS)
\( c[n] = c[n-1] + \mu X'[n] e[n] \)
\( (\mu > 0) \)

typical choices, \( (\mu = 1, e = 0) \)

now we can easily select standard values of \( \mu \) for various different random processes.

- Sign LMS:
\( c[n] = c[n-1] + 2\mu X'[n] \text{sign}(e[n]) \)
\( \text{sign}(a+ib) = \text{sign}(a) + i\text{sign}(b) \)

this is much less computationally complex version.

- More complexity:
\( \begin{array}{c|c|c|c|c|c} \text{Sign} & \text{LMS} & \text{NLMS} & \text{Recursive} & \text{More complex} \\ \hline \end{array} \)

lets start the analysis for the deterministic algorithm - the SDA.
SDA: $C_k = C_{k-1} + \gamma u (R_{xx} C_0 - R_{xx} C_{k-1})$
$-x_0 - C_0$

the error in the estimate of the optimal filter:

$\hat{e}_k = \hat{e}_{k-1} - \gamma u R_{xx} \hat{e}_{k-1} = (I - \gamma u R_{xx}) \hat{e}_{k-1}$

$R_{xx} = Q \Lambda Q^H$. $\hat{C}_k = Q^H \hat{e}_k$ $(Q^H Q = I = QQ^H)$

$\Lambda = \begin{pmatrix} \lambda_1 & 0 \\ 0 & \lambda_2 \end{pmatrix}, \lambda_i > 0$

well as

$\hat{C}_k = (I - \gamma u \Lambda) \hat{C}_{k-1}$

now the transition matrix $(I - \gamma u \Lambda)$ is diagonal

$\begin{bmatrix} \hat{C}_{k,11} \\ \vdots \\ \hat{C}_{k,n} \end{bmatrix} = \begin{bmatrix} 1 - \gamma u \lambda_1 & 0 \\ \vdots & \vdots \\ 0 & 1 - \gamma u \lambda_n \end{bmatrix} \begin{bmatrix} \hat{C}_{k-1,11} \\ \vdots \\ \hat{C}_{k-1,n} \end{bmatrix}$

no coupling.

so we can show for an arbitrary index of $\hat{C}_k$ show it converges to zero.

$\hat{C}_{k,1} = (1 - \gamma u \lambda_1) \hat{C}_{k-1,1} = (1 - \gamma u \lambda_1)^2 \hat{C}_{k-2,1} = \ldots$

$= (1 - \gamma u \lambda_1)^k \hat{C}_{0,1}$

(time $k \to \infty$)

$\hat{C}_{k,1} \to 0$ as long as $|1 - \gamma u \lambda_1| < 1$

so we need to select $\gamma > 0$ to be small enough to ensure this convergence.

we need this $\forall I$. 
How to select $\mu$:
we want $-1 < 1 - 2\mu \lambda_k < 1$
we already require $\mu > 0 \rightarrow$ that $\mu$ makes sure $1 - 2\mu \lambda_k < 1$
(since $\lambda_k > 0$).
we have to make sure $1 - 2\mu \lambda_k > -1$.
so we can just make sure this holds for $\lambda_{\max}$.
lets suppose they are sorted: $\lambda_{\max} = \lambda_1$, $\lambda_{\min} = \lambda_{n}$.
so we need $1 - 2\mu \lambda_{\max} > -1$
so $\mu < \frac{1}{\lambda_{\max}}$
so pre-bounds for $\mu$: $0 < \mu \leq \frac{1}{\lambda_{\max}}$
conservative $\rightarrow$ aggressive

We should also regard $\lambda_{\min}$:
lets suppose were really aggressive and select $\mu = \frac{1}{\lambda_{\max}}$
Then:

$\Delta - 2\mu \lambda_{\min} = 1 - 2 \frac{1}{\lambda_{\max}} \lambda_{\min}$
the last mode will be the bottleneck - take long

time to converge.

$\lambda_{\min}$ is a good measure of convergence potential.

Condition number for $R : \gamma(R) = \frac{\lambda_{\max}}{\lambda_{\min}}$
we can select with heuristic making the highest
and lowest modes equal in magnitude

$1 - 2\mu \lambda_{\min} = -(1 - 2\mu \lambda_{\max}) \rightarrow \mu = \frac{1}{\lambda_{\min} + \lambda_{\max}}$

Examine the MSE:
$P(C_k) = P_0 + C_k^* R_{\lambda k} C_k = P_0 + C_k^* \Lambda C_k$
\[ P_0 + \sum_{k=1}^{M} \lambda_k \left( \sum_{t=1}^{k} a_{t} \right)^2 = P_0 + \sum_{k=1}^{M} \lambda_k \left( \frac{14 - 2a a_{k}}{1} \right)^2 \left( \sum_{t=1}^{k} a_{t} \right)^2 \]

the MSE converges a bit faster than the weights \( \lambda_k \) themselves.

A general statement:

\[ P(\lambda_k) \leq \left( \frac{Y(R)-1}{Y(R)+1} \right)^2 P(\lambda_{k-1}) \]

this is a bound to the convergence rate.

now let's analyse the stochastic version: the LMS:

\[ C(n) = \lambda \left( C(n-1) + 2 \right) x[n] e[n] \]

how does the LMS converge behave?

\( C(n) \) is now a random process itself.

\( C(n) \) cannot converge to a constant (since \( \mu > 0 \) and we keep updating).

- If we want \( C(n) \to \text{const} \) we have to use changing \( \mu : \mu(n) \to 0 \)

How should \( \mu(n) \) go to zero.

- for non-stationary case we may want \( \mu(n) \to \mu \) small but not zero.

a common approach:

Stochastic Approximation Methods
\[ \sum_{n=1}^{\infty} \mu_n = \infty \quad \mu_n \to 0 \quad \text{(for instance, } \mu_n = \alpha \frac{1}{n} \text{)} \]

\[ E(c_n) \to \rho \quad \text{how to select } \mu \text{ for that?} \]

\[ \text{Covar}(c_n) = 0 \to \text{small values} \]

It turns out that if selecting \( 0 < \mu \leq \frac{1}{\lambda_{\text{max}}} \) we \( \to \) get \( E(c_n) \to \rho \).

However, this condition is not enough to satisfy the result on covariance.

\[ \text{Feur & Weinstein (Feb 85)} \]

\[ c_{n+1} = c_n - \alpha + 2\mu x_n^* y_n - c_n^{(n)} x_n^* y_n \]

\[ = \left( I - 2\mu x_n^* y_n^* \right) c_n + 2\mu x_n^* y_n \]

\[ E(c_{n+1}) = \left( I - 2\mu R_{x_n} \right) E(c_n) + 2\mu E(x_n^* y_n) \]

\[ (\text{we would like to separate } x_5 \text{'s from } c) \]
Now let's analyse the LMS algo:
\[ c[n] = c[n-1] + 2\mu \times x[n] e^*[n] \quad (c[n] = y[n] - c[n-1] x^*[n]) \]

the LMS is not an order-recursive procedure, so we can simply write \( x[n] \) instead of \( x[n-1] \). M is fixed throughout the process.

Examine mean & var of \( c[n] \)

Start in the last case, we want \( E[c[n]] \rightarrow c_0 \)

and var \( \rightarrow 0 \), something small, bounded.

\[ c[n] = c[n-1] + 2\mu \times x[n] (y[n] - c[n-1] x^*[n]) = (I - 2\mu \times x[n] x^*[n]) c[n-1] + 2\mu \times x[n] y[n] \]

Subtract optimum \( c_0 \) from both sides and rearrange:
\[ \left( E[c[n]] = y[n] - c_0 x^*[n] \right) \]

\[ E[c[n]] = E[y[n]] (I - 2\mu \times x[n] x^*[n]) c[n-1] + 2\mu \times x[n] (E[y[n]] c_0 + 2\mu \times x[n] E[e]^*[n]) \]

\[ E[c[n]] = (I - 2\mu \times x[n] x^*[n]) c[n-1] - (I - 2\mu \times x[n] x^*[n]) c_0 + 2\mu \times x[n] E[e]^*[n] \]

\[ \tilde{E}[c[n]] = (I - 2\mu \times x[n] x^*[n]) \tilde{E}[c[n-1]] + 2\mu \times x[n] E[e]^*[n] \]

Let's analyse this equation

Orthogonality:
\[ E(c[n]) = E((I - 2\mu \times x[n] x^*[n]) \tilde{E}[c[n-1]] + 0) \]
in general, $\mathcal{E}(n-\delta)$ depends on $y_\delta, \ldots, y_{n-1}, \mathcal{E}(n-2) \ldots \mathcal{E}(1)$ and recursively on all the past data. So in general we cannot separate the $E(\cdot)$ to $E(\cdot) \cdot E(\cdot)$. In general $\mathcal{E}(n-\delta)$ is correlated with $x_{n-1}$.

Simplify by assumption:

A1) $[y_\alpha(n)]$ is an iid sequence $\Rightarrow \mathcal{E}(n-\delta)$ is independent of $x(n)$

Clearly this assumption is not relevant for time series data $x(n)$.

So now we can split:

$E(\mathcal{E}(n)) = E(I - 2\mu x(x_{\alpha}(n))) \cdot E(\mathcal{E}(n-\delta)) = (1 - 2\mu)E(\mathcal{E}(n-\delta))$

This is the same as for the SDA.

So here also we need $0 < \mu < {\lambda}_{max}$ for convergence of $E(\mathcal{E}(n))$ to zero.

The convergence rate depends on $1 - 2\mu\lambda_{max}$

We want to minimize $\max_{\lambda} |1 - 2\mu\lambda_{max}|

1 - 2\mu\lambda_{max} < \cdots < 1 - 2\mu\lambda_{min}$

A good choice is to have $1 - 2\mu\lambda_{max} = -(1 - 2\mu\lambda_{min})$

and getting $\mu = \frac{1}{2\lambda_{max}}$.

Now analyse $\mathcal{E}(n) = E(\mathcal{E}(n)) \cdot \mathcal{E}(n)$ $\mathcal{E}(n) \rightarrow \infty$? Still on the same case, and still under assumption A1) $\left(\frac{y_\alpha(n)}{x(n)}\right)$ iid
why do we want this to be small:
1) leads to tighter bounds on $\mu$
2) builds confidence on LMS.
3) Insight into the excess mean squared error.

$$P(n) = E\left(\left|\text{err}_n\right|^2\right) = E\left(\frac{1}{m} - c^T_{\text{err}} \hat{\theta}_n \times \text{err}_n^2\right) =$$

$$= E\left(1 + c^T_{\text{err}} \hat{\theta}_n \times \text{err}_n^2\right) - c^T_{\text{err}} \times \text{err}_n^2 =$$

$$= E\left(1 + c^T_{\text{err}} \hat{\theta}_n \times \text{err}_n^2\right) - E\left(\text{err}_n \times \text{err}_n^2\right) =$$

$$= E\left(1 + c^T_{\text{err}} \hat{\theta}_n \times \text{err}_n^2\right) - E\left(\text{err}_n \times \text{err}_n^2\right) =$$

$$= E\left(\text{err}_n \times \text{err}_n^2\right)$$

$$\Rightarrow$$

$$E\left(\text{err}_n \times \text{err}_n^2\right) = E\left(\text{err}_n \times \text{err}_n^2\right)$$

where $E\left(\text{err}_n \times \text{err}_n^2\right)$ is the $i^{th}$ diagonal element of $\Theta$. $\Theta$ is a vector of these values.

$$\lambda = \begin{bmatrix} \frac{n}{m} \\ \vdots \\ \frac{n}{m} \end{bmatrix}$$
so \( P_{\alpha, \beta} = \text{tr}(R_{\alpha, \beta} R_{\beta, \alpha}) = \text{tr}(\Lambda_{\alpha, \beta}) = \lambda_{\alpha, \beta} \)

Let's add another assumption:

A2) \((\mathbf{y}_{\alpha}, \mathbf{x}_{\alpha})\) is a circular Gaussian random vector, zero-mean.

Circular Gaussian:

\[
\mathbf{z} = \mathbf{x} + j \mathbf{y}, \quad E(\mathbf{z}) = 0 \implies E(\mathbf{x}) = 0, \quad E(\mathbf{y}) = 0
\]

\(\mathbf{x}, \mathbf{y}\) jointly Gaussian

\[
E(\mathbf{x}\mathbf{x}^H) = E(\mathbf{y}\mathbf{y}^H)
\]

\[
E(\mathbf{x}\mathbf{x}^H) = 0
\]

This means:

\[
E(\mathbf{z}\mathbf{z}^H) = E(\mathbf{x}\mathbf{x}^H) + E(\mathbf{y}\mathbf{y}^H)
\]

\[
E(\mathbf{z}\mathbf{z}^H) = 0 \text{ matrix}
\]

\[
\mathbf{A}^H = (I - 2\mu \mathbf{x}(\alpha) \mathbf{x}(\alpha)^H), \quad \mathbf{B} = 2 \mu \mathbf{x}(\alpha) \mathbf{e}_\alpha^H \mathbf{x}(\alpha)
\]

\[
E(\mathbf{z}(\alpha)\mathbf{z}(\alpha)^H) = E((\mathbf{A} + \mathbf{B})(\mathbf{A}^H + \mathbf{B}^H)) = E(\mathbf{A}\mathbf{A}^H + \mathbf{B}\mathbf{B}^H + \mathbf{A}\mathbf{B}^H + \mathbf{B}\mathbf{A}^H)
\]

\[
E(\mathbf{A}\mathbf{B}^H) = E((I - 2\mu \mathbf{x}(\alpha)^H \mathbf{x}(\alpha)) \mathbf{z}(\alpha) \mathbf{z}(\alpha)^H E(\mathbf{A}) \mathbf{e}_\alpha \mathbf{e}_\alpha^H)
\]

\[
E(\mathbf{A}\mathbf{B}^H) = \mathbf{0} = E(\mathbf{B}\mathbf{A}^H)
\]

\[
E(\mathbf{B}\mathbf{B}^H) = 4 \mu^2 E(\mathbf{x}(\alpha) \mathbf{x}(\alpha)^H \mathbf{e}_\alpha \mathbf{e}_\alpha^H) = 4 \mu^2 \mathbf{R}_\mathbf{x}
\]

\[
E(\mathbf{A}\mathbf{A}^H) = E((I - 2\mu \mathbf{x}(\alpha)^H \mathbf{x}(\alpha)) \mathbf{z}(\alpha) \mathbf{z}(\alpha)^H (I - 2\mu \mathbf{x}(\alpha)^H \mathbf{x}(\alpha)) - \mathbf{C}(\alpha - \mathbf{z}(\alpha))^H \mathbf{C}(\alpha - \mathbf{z}(\alpha))^H \mathbf{R}_\mathbf{x} \mathbf{x}(\alpha)^H \mathbf{R}_\mathbf{x} \mathbf{x}(\alpha)^H
\]

\[
E(\mathbf{C}(\alpha - \mathbf{z}(\alpha))^H \mathbf{C}(\alpha - \mathbf{z}(\alpha))^H \mathbf{R}_\mathbf{x} \mathbf{x}(\alpha)^H \mathbf{R}_\mathbf{x} \mathbf{x}(\alpha)^H)
\]

\[
\mathbf{C}(\alpha - \mathbf{z}(\alpha))^H \mathbf{C}(\alpha - \mathbf{z}(\alpha))^H \mathbf{R}_\mathbf{x} \mathbf{x}(\alpha)^H \mathbf{R}_\mathbf{x} \mathbf{x}(\alpha)^H
\]
\[ \phi(t) = \phi(0) \exp \left( -\mu t \right) - 2 \mu \int_0^t \phi(s) \, ds - 2 \mu \int_0^t \phi(s) \, ds + 4 \mu^2 R \phi(t) R + \frac{4 \mu^2 R \phi(t) R}{C(BB^T)} \]

so now we have a deterministic recursive formula for \( \phi(t) \).

We want to prove that \( \phi(t) \rightarrow \phi(0) \) constant matrix \( R = \Phi \) \( Q^T Q = Q^T Q = I \)

\[ \Theta(t) = Q^T \phi(t) \]

Thus, multiply \( Q^T \) to the recursive equation.

\[ Q \Theta(t) = -\Theta(t) - 2 \mu Q \Theta(t) + 2 \mu Q \Theta(t) + 4 \mu^2 \Lambda R \phi(t) R + 4 \mu^2 \Lambda R \phi(t) R \]

we can look at just the diagonal values (we saw that is enough): go to vector \( Q \Theta(t) \).

\[ Q \Theta(t) = \Theta(t) - 2 \mu Q \Theta(t) + 4 \mu^2 \Lambda Q \Theta(t) + 4 \mu^2 \Lambda R \phi(t) R + 4 \mu^2 \Lambda R \phi(t) R \]
back to the LMS algo' \( (w_{s,s} \text{ signal}) \)
\[
C_{n+1} = C_{n} - 2\mu E(x_n e^*_n)
\]

\[
E(C_{n+1}) = E(C_{n}) - 2\mu E(x_n e^*_n)
\]

\[
E(C_{n}) \to C_0 \text{ with } o<\mu < \frac{1}{\gamma_{\text{max}}}
\]

we can also see this limit \( C_0 \)

since in the limit we would expect

orthogonality principle, meaning \( E(x_n e^*_n)=0 \)

then \( E(C_{n}) = E(C_{n-1}) \) (convergence)

\[
\hat{\mu} = 2\mu
\]

\[
\hat{C}(n) = (I - \hat{\mu} x(n)x^H(n))\hat{C}(n-1) + \hat{\mu} x(n)e^*_n
\]

\[
(C_{n+1} = y(n) - c^*_n x(n))
\]

we had 3 assumptions:

A1) \([x(n)] \text{ is an iid sequence}

A2) \( \text{Circular Gaussian & zero mean, } \)

(for fourth moment calculation + \([\text{uncorrel} \Rightarrow \text{indep}])

\[
\phi_{n} = E(\hat{C}(n)\hat{C}^*_n) \quad P_{n} = E(\phi(n)^2) = p_0 + P_{xn}(n)
\]

\[
\text{P}_{xn}(n) = \text{tr} R \phi_{n-1} = \text{tr} \Lambda \phi_{n-1} = \Lambda^T [\phi_{n-1}]
\]

\[
\phi_{n-1} = Q^* \phi_{n-1} Q \quad (R = Q \Lambda Q^*)
\]

\[
\Lambda = \text{diag} (\Lambda_{n-1})
\]
\[ \Phi(n) = E(\Phi(n)\Phi^H(n)) = E(\Phi(n)) + E(\Phi^H(n)) + E(\Phi(n)\Phi(n)) + E(\Phi^H(n)\Phi^H(n)) \]
\[ = E(\Phi(n)) + \tilde{\mu}^2 R \Phi_0 = E(\tilde{\Phi}(n-1)\tilde{\Phi}^H(n-1)) + \tilde{\mu}^2 E(x(n)x(n)\tilde{\Phi}(n-1)\tilde{\Phi}^H(n-1)) \]
\[ - \tilde{\mu} E(x(n)x(n)\tilde{\Phi}(n-1)) - \tilde{\mu} E(\tilde{\Phi}(n-1)x(n)\tilde{\Phi}^H(n-1)) + \tilde{\mu}^2 R \Phi_0 \]
\]

the second term (the most complicated one) results in \( R \Phi_0 x(n) + R \tilde{\Phi}(n-1) \). in order to get it we use the property:
\[ E(z_1, z_2, z_3, z_4) = E(z_1, z_2) E(z_3, z_4) + E(z_1, z_4) E(z_2, z_3) \]
for \( z_{1,4} \) jointly complex circular Gaussian zero mean.

we get the recursive formula for \( \Phi(n) \):
\[ \Phi(n) = \Phi(n-1) - \tilde{\mu} \Phi(n-1) R - \tilde{\mu} R \Phi(n-1) + \tilde{\mu}^2 R \Phi_0 x(n) + \tilde{\mu}^2 R \Phi_0 x(n) + \tilde{\mu}^2 R \Phi_0 x(n) + \tilde{\mu}^2 R \Phi_0 x(n) \]

now let's do the recursive formula for \( \Theta(n) = \Phi(n) \Theta(n) \):
\[ \Theta(n) = \Theta(n-1) - \tilde{\mu} \Theta(n-1) \tilde{\Phi}(n-1) + \tilde{\mu} \tilde{\Phi}(n-1) \Theta(n-1) + \tilde{\mu} \tilde{\Phi}(n-1) \Theta(n-1) + \tilde{\mu} \tilde{\Phi}(n-1) \Theta(n-1) \]
\[ (Q^n (R \tilde{\Phi}(n-1)) Q = Q^n (R \tilde{\Phi}(n-1)) Q \]

now for the vector \( \Theta(n) \): (taking the diag)
\[ \Theta(n) = \Theta(n-1) - \tilde{\mu} \tilde{\Phi}(n-1) \Theta(n-1) + \tilde{\mu} \tilde{\Phi}(n-1) \Theta(n-1) + \tilde{\mu} \tilde{\Phi}(n-1) \Theta(n-1) + \tilde{\mu} \tilde{\Phi}(n-1) \Theta(n-1) \]
\[ = (I - \tilde{\mu} \tilde{\Phi}(n-1) \Theta(n-1)) \Theta(n-1) + \tilde{\mu} \tilde{\Phi}(n-1) \Theta(n-1) + \tilde{\mu} \tilde{\Phi}(n-1) \Theta(n-1) \]
\[ = B \Theta(n-1) + \tilde{\mu} \Phi_0 x(n) \]
what is matrix B:

$$B = \begin{pmatrix} (4 - \lambda^2) & 0 \\ 0 & 1 - \lambda^2 \end{pmatrix} + \lambda^2 A^T$$

for convergence of $\text{Markov}$ we need:

$$|\lambda_i(B)| < 1$$

we already know that we have to enforce

$$\mu < \frac{1}{\lambda_{\text{max}}}$$

to get convergence of mean,

now we need to make sure covar also converges.

It turns out we need:

$$\frac{\sum_{x=1}^{M} \lambda_{\text{mix}}}{\sum_{x=1}^{M} 2 - \lambda_{\text{mix}}} < 1$$

(necessary & sufficient)

this implies $\mu$ has to be much smaller than $\frac{1}{\lambda_{\text{max}}}$.

Assume $\mu$ has been chosen properly.

Let's check what happens to $\theta^{(t)} \overset{\infty}{\rightarrow} O(\infty)$

take limit on both sides of recursive equation:

$$O(\infty) = B \cdot O(\infty) + \lambda^2 A^T P_0 \rightarrow$$

$$O(\infty) = \left( \lambda^2 P_0 (I - B)^{-1} \right) \lambda$$

$$P_{\text{ex}}(\infty) = \pi^T O(\infty) = \lambda^2 P_0 \pi^T (I - B)^{-1} \lambda$$
remember B also depends on selection of \( \mu \).

For small (close to zero) \( \mu \):

\[
B \times \begin{bmatrix}
1 - 2\tilde{\lambda} + c \\
0 \\
0 \\
1 - 2\tilde{\lambda} + c
\end{bmatrix} + 0 = I - 2\tilde{\lambda} \Lambda
\]

So \( I - B \approx 2\tilde{\lambda} \Lambda \) (small \( \mu \))

So: \( \text{Pex}^{[00]} \approx \frac{\mu^2}{3} \Lambda^2 \sum_{i=1}^{\infty} x_i \approx \text{Pex}^{[00]} \)

\[
\text{Pex}^{[00]} \approx \mu \cdot \text{tr} \left( \text{R} \right)
\]

The bound for convergence in mean square is much stricter (harder) than for convergence in mean.

Notice: matrix B is not just PSD, it is even more special - all its entries are positive.

This implies some properties. For instance:

- \( \lambda_{\text{max}}(B) \) is unique (all other eigenvalues are smaller)
- and the corresponding eigenvector has all entries with the same sign.

An alternate expression for \( \text{Pex}^{[00]} \):

\[
\Theta^{[00]} = B \Theta^{[00]} + \hat{\mu}^2 \text{P}_0.
\]

\[
(\text{I} - \begin{bmatrix}
(1 - 2\tilde{\lambda})x_1 & -x_1 \\
0 & (1 - 2\tilde{\lambda})x_1 \\
2\tilde{\lambda} - x_1^2 & 0 \\
0 & 2\tilde{\lambda} - x_1^2
\end{bmatrix}) \Theta^{[00]} = \hat{\mu}^2 \text{P}_0
\]

\[
\Theta^{[00]} = \frac{\hat{\mu}^2}{\lambda_{\text{max}}} \left( \text{Pex}^{[00]} + \text{P}_0 \right)
\]
\[ q[n] = \mu \left( \begin{array}{c}
\frac{1}{2-\lambda_1^2} \\
\vdots \\
\frac{1}{2-\lambda_m^2} 
\end{array} \right) (P_{\text{ex}(\infty)} + P_0) \]

\[ P_{\text{ex}(\infty)} = \sum_{k=1}^{\infty} \frac{\lambda_k^2}{2-\lambda_k^2} (P_{\text{ex}(\infty)} + P_0) = \gamma (P_{\text{ex}(\infty)} + P_0) \]

\[ \Rightarrow \left[ P_{\text{ex}(\infty)} = \frac{\gamma P_0}{1-\gamma} \right] = \frac{\gamma}{2} (1-B)^{1+s} \]

here we can see that \( \gamma < 1 \) is the requirement for mean squared convergence.

Let's examine the eigenvalues of \( B \) (only interested in \( n \)) largest eigenvalue

\[ B_g = c^g \]

we know \( c < 1 \) for small \( \mu \).

try to increase \( \mu \) until we hit \( \max (\beta) = 1 \) and that will be our boundary.

\( j \)th entry of \( B_g \)

\[ (1-\lambda g_j)^2 g_j + \lambda^2 \sigma^2 \sigma^t g_j = c \cdot g_j \]

\[ g_j = \frac{\lambda^2 \sigma^2 \sigma^t g_j}{c - (1-\lambda g_j)^2} \]

the denom is always pos. \( \lambda^2 \sigma^2 \sigma^t \) is pos.

whatever the sign of \( \lambda^2 \sigma^t \) - this will be the sign of \( g_j \) \( \forall i = 1, ..., m \)
Critical $\mu$ value: 

$$\mu_{\text{critical}} \rightarrow c = 1$$

\[ \Delta_l = \frac{\beta \mu - 2 R_l}{1 - (1 - \beta R_l) 2 R_l} = \frac{\beta \mu - 2 R_l}{1 - (1 - \beta R_l) 2 R_l} \]

\[ \Delta_l = \frac{\mu R_l}{1 - (1 - \beta R_l) 2 R_l} \]

Now, analyze LMS for tracking non-stationary signals.

This time, we'd like $\mu$ large enough to be able to keep track with changes.

Assumptions:

1) $X[n]$ is i.i.d., circular Gaussian, mean zero, covar $R$
2) $\psi[n] = C_{\psi}^{-1/2} X[n] + e_{\psi}[n]$ (zero mean, win $\psi[n]$, Gaussian)
3) $C_{\psi}[n] = \rho_{\psi}[n-1] + 4\psi[n]$ (another win $\nu_\psi[n]$)
   Assume $\rho = 1$ for analysis (random walk model)
4) $X[n], e_{\psi}[n], X[n]$ are all independent

Notice $X[n]$ is still stationary. The non-stationarity in $Y$ comes from the changing $C_{\psi}[n]$.
\textbf{The LMS Algorithm Update:}
\[ C(n) = C(n-1) + \hat{\alpha} X(n) e^*_{(n)} \]

\[ \hat{\alpha}^n = C(n) - C_{0, n} = C(n-1) + \hat{\alpha} X(n) e^*_{(n)} - C_{0, n} \]

\[ = C(n-1) + \hat{\alpha} X(n) (e_{(n)} - \hat{\alpha}^n X(n) e^*_{(n)} - C_{0, n-1} + Y_{(n)}) \]

\[ = (I - \hat{\alpha} X(n)^H e^*_{(n)}) \hat{\alpha}^n X(n) + \hat{\alpha}^n X(n) e^*_{(n)} - Y_{(n)} \]

\[ = \hat{e}^n_{(n)} \]

\textbf{Analysis of the Coefficient Error:}
\[ E(\hat{e}^n_{(n)} e^*_{(n)}) = \phi(n) = E(\alpha^2) + E(\beta^2) + E(\alpha \beta) + E(\beta \alpha^*) \]

(\phi(n) is independent of all other terms and has zero mean)
\[
\Phi_{\text{n-1}} - \lambda R \Phi_{\text{n-1}} - \mu \Phi_{\text{d-1}} + \lambda R \Phi_{\text{d-1}} + \mu^2 R P_0 x_{00} + \mu^2 R P_0 + \mu^3 I
\]

Let's diagonalize again \( R = Q \Lambda Q^T \) (like before)

\[
\Lambda^{1/2} Q^T Q = \Theta[\Lambda]^{1/2} \quad \text{(like before)}
\]

\[
\Theta[\Lambda] = B \Theta[\Lambda] + \mu^2 P_0 + \mu^3 I
\]

\[
\Theta[\Lambda] = (I - B)^{-1} \left[ \Lambda^{1/2} P_0 + \Lambda^{1/2} I \right]
\]

\[
P_{\text{ex}}[\Lambda] = \Lambda^{1/2} P_0 [2 (I-B)^{1/2} + \mu^2 (I-B)^{1/2}] \] (neglect small terms)

For small \( \mu \): \( I - B \approx \mu \Lambda \)

\[
P_{\text{ex}}[\Lambda] = \frac{\mu^2 P_0}{\mu} \Lambda^{1/2} + \mu^2 \Lambda^{1/2} I
\]

we see the tradeoff: we want \( P_0 \) or \( \mu \) small.
the first term - we want small \( \hat{x} \) (to converge to correct \( x_0 \)), the second term - we want large \( \hat{x} \) (to track changes).

There's a tradeoff. if \( \hat{x} \) is too small - then tracking error is too large.

prof wants to decide what we learn next.

Recursive Least Squares: 
\[
C(n) = y(n) - \hat{x}(n) \times \hat{y}(n)
\]
so far we have minimized 
\[
E(\hat{e}(n)^2)
\]
\[
J = \sum_{n=0}^{N} |e(n)|^2
\]
now minimize \( J \), no need to explicitly assume \( \hat{x}_0, \hat{y}_0 \) are random processes.

Advantage: general and widely used.

Disadvantage: no model for non-stationary case.

We can have an update per time varying variables, using weighting of samples, disadvantage: need a lot of prior information.

Kalman Filter:

\[
C_{n+1} = F_n C_n + G n U_n + G n W_n
\]
(system dynamics) \( \rightarrow \)
\[
y_{n+1} = h_m C_{n+1} + V_n \]
(linear equation)

Best estimate of \( c_{n+1} \) given \( y_{n+1} \), \( m \leq n-d \) (prediction)

Best estimate of \( c_{n+1} \) given \( y_{n+1} \), \( m = n \) (filtering)

Advantage: optimal (under model assumptions), can handle many

In the stationary case, we can reduce Kalman filter to stationary Least squares prob.
prof decided to take the least squares way

\[ e(t) = y(t) - c^T x(t) \]
\[ y(t) = c^T x(t) + e(t) \]
\[ y^*(t) = x^T c + e^*(t) \]

\[
\begin{bmatrix}
    y^*(0) \\
    y^*(1) \\
    \vdots \\
    y^*(N-1) \\
\end{bmatrix}
= \begin{bmatrix}
    x^T(0) \\
    x^T(1) \\
    \vdots \\
    x^T(N-1) \\
\end{bmatrix} - \begin{bmatrix}
    e^*(0) \\
    e^*(1) \\
    \vdots \\
    e^*(N-1) \\
\end{bmatrix}
\]

\[
y = Xc + e\]

\[ \min\|e\|^2 = \min_c \sum_{n=0}^{N-1} |e(n)|^2 \]

No class Next week.

Linear Least Squares \(\rightarrow\) Recursive Least Squares

\[
\begin{bmatrix}
    y(0) \\
    y(1) \\
    \vdots \\
    y(N-1) \\
\end{bmatrix}
= \begin{bmatrix}
    c^T x(0) \\
    c^T x(1) \\
    \vdots \\
    c^T x(N-1) \\
\end{bmatrix}
\]

\[
\begin{bmatrix}
    e^*(0) \\
    e^*(1) \\
    \vdots \\
    e^*(N-1) \\
\end{bmatrix}
= \begin{bmatrix}
    y^*(0) - x(0)c \\
    y^*(1) - x(1)c \\
    \vdots \\
    y^*(N-1) - x(N-1)c \\
\end{bmatrix}
\]

\[ c = y - Xc \]
\[ \operatorname{Min} \sum_{c} \{ \| e_{c} \|_{2}^{2} \} = \min_{e} \mathbb{E}[e^{2}] = \min_{e} \mathbb{E}[e_{1}^{2}] \]

This problem is general: \( n \) doesn't have to be time, it can be some index of instances drawn from a sample.

\[ y \in \mathbb{R}^{n}, \quad x \in \mathbb{R}^{m}, \quad c \in \mathbb{R}^{m+1} \]

Cases:

- \( n < m \) underdetermined, we have less measurements (constraints) than unknowns (params).
- \( n = m \) then we can get zero error with infinite solutions.
- \( n > m \) overdetermined, we have more constraints than parameters.

We can describe \( x \) by its columns: \( X = [x_{1}, \ldots, x_{m}] \) and we'll assume \( x_{1}, \ldots, x_{m} \) linearly independent, then \( \operatorname{rank}(X) = m \).

Objective - cost function:

\[ J(c) = e^{H} e = (y - Xc)^{H} (y - Xc) = y^{H} y + c^{H} X^{H} c - y^{H} x_{c} \]

\[ 2^{*} J = X^{H} c - X^{H} y \quad \text{we get:} \]

\[ x^{H} x c \_{ls} = x^{H} y \]  

(reminds \( Rx \cdot c = R y \) in Levinson)

When \( x \) is full rank \( x^{H} x \) (\( m \times n \)) is invertible

\[ c_{ls} = \left( x^{H} x \right)^{-1} x^{H} y \]
Recursive LS: we want to get the next \( C_{ls} \) (for next sample \( n \)) -- but without too much computation -- just update the \( C_{ls} \)

plus -- we want this to handle a non-stationary sequence of samples -- use weighting of instances to handle it (latest instances get more weight)

\[
J(C_{ls}) = y^H y - C_{ls}^H X^H y = y^H y - y^H X C_{ls}
\]

\[
C_{ls} = y - X C_{ls}
\]

\( y^H x \) is a linear combination of \( X \)'s columns, we want this to approximate \( y \) (in least square sense)

\[
R(x) = \text{Range space of } X = \{ z \in \mathbb{R}^n | z = X \beta \}
\]

supspace of \( \text{dim } M \)

claim: \( C_{ls} \perp R(x) \)

meaning: \( C_{ls} \perp X \beta \), \( \forall \beta \)

\[
C_{ls}^H X \beta = (y - X C_{ls})^H X \beta = 0
\]

\[
(y^H x - C_{ls}^H x^H) \beta = (x^H y - X^H C_{ls}) \beta = 0
\]

\[
\hat{y}_{ls} = X C_{ls} = X (X^H X)^{-1} X^H y = p_x y
\]

\[
\hat{y}_{ls} = p_x y
\]

\[
C_{ls} = y - \hat{y}_{ls} = (I - p_x) y
\]
\[ P_x = X(X^*X)^{-1}X^* \]

- \( P_x \) is an orthogonal projection matrix
- \( P_x = P_x^* \) (hermitian symmetric)
- \( P_x = P_x P_x = P_x^2 \) (if you projected once, another projection will not change anything)
- \( P_x^\perp = (I - P_x) \) is projection to the orthogonal complement of \( R(x) \)
- \( \forall z \in X = P_x z + P_x^\perp z \)

\[
\begin{bmatrix}
1 & 0 \\
0 & 1
\end{bmatrix}
\quad,
\begin{bmatrix}
2 & 1 \\
1 & -2
\end{bmatrix}
\] both these \( X \) matrices span the same space, hence have the same orthogonal projection matrix.

We'd prefer working with the first one, since it has orthogonal basis vectors as columns.

\[ J(c_{1s}) = \| e_{1s} \|^2 = e_{1s}^* e_{1s} = (P_x^\perp y)^* (P_x^\perp y) = y^* P_x^\perp P_x^\perp y = y^* P_x^\perp P_x y \]

---

**Two Applications:**

1. NLMS
2. Forward Backward LP

**1) Normalized LMS:**

We have \( c_{1}(n) \), new sample arrives \([y(n)]\), we want to update to \( c_{1}(n) \).

\[ c_{1}(n) = y(n) - C_{1s-n}^* x(n) \] \( \text{approx. error} \)

\[ c_{2} = c_{1}(n-1) + \Delta c \] \( \text{we want } \Delta c \text{ such that } e_{2s} \) (with \( e_{2s} \) will now be zero)

\[ e_{2s} = y(n) - C_{1s-n}^* x(n) \] \( \text{posterior error (after correct)} \)

\[ = y(n) - C_{1s-n}^* x(n) = \Delta c^* x(n) \]
\[ e_0(n) = e_0(n) - \Delta c^n x(n) \]
choose \( \Delta c \) to make \( \| e_0(n) \| \) as small as possible
this is underdetermined - we can make \( \| e_0(n) \| \) be zero in many possibilities (4 equation, 3 unknowns)
infinite choices for \( \Delta c \).
so let's introduce another requirement
we want a small perturbation \( \Delta c \).

\[
\min \| \Delta c \|_2^2 \quad \text{s.t.} \quad e_0(n) = \Delta c^n x(n)
\]
(or \( e_0(n) = x^t(n) \Delta c \))
solution is \( \Delta c = \alpha \frac{x(n)}{\| x(n) \|_2} \)
we get:
\[
e_0^*(n) = \alpha \frac{x^t(n) x(n)}{\| x(n) \|_2^2} = \alpha \frac{\| x(n) \|_2^2}{\| x(n) \|_2^2}
\]
\[
\alpha = \frac{\| x(n) \|_2}{\| x(n) \|_2}
\]
so \( \Delta c = \frac{e_0^*(n) x(n)}{\| x(n) \|_2^2} \)

\[
c(n) = c(n-1) - \frac{x(n) e_0^*(n)}{\| x(n) \|_2^2}
\]
To modify, more 'safe solution':
\[
c(n) = c(n-1) + \mu \frac{x(n) e_0^*(n)}{\| x(n) \|_2^2 + \varepsilon}
\]
(\( \varepsilon > 0 \), \( 0 < \mu < 1 \))

2) Forward, Backward Linear prediction:

going back to time series data, one step prediction.
final pred: \( e_0(n) = x(n) + \sum_{i=1}^{p} a_i e_0(n-i) \)
we have data \( x(n) \) \( n = 0 \ldots N-1 \)
options: we can start with \( e_0(n) \)
we can zero-pad and start at \( e_0(0) \).
\[
\begin{bmatrix}
L_x^*(a) \\
L_x^{*(m-1)} \\
\vdots \\
L_x^{*(m-k)} \\
L_x^{*(m-k+1)} \\
\end{bmatrix} = \begin{bmatrix}
x^{*m-1}, 0, 0, \ldots, 0 \\
x^{*(m-2)}, \ldots, x^{*0}, 0 \\
\vdots \\
x^{*(m-k)}, \ldots, x^{*(m-k+1)}, 0 \\
0, x^{*(m-k+1)}, \ldots, x^{*(m-k+1)} \\
\end{bmatrix} = \begin{bmatrix}
a_0 \\
a_1 \\
\vdots \\
a_{m-1} \\
0 \\
\end{bmatrix}
\]

(it's like \( e = (y, -x)(e_c) \))

can solve for \( a_m \) using Least squares.

we want to minimize
\[
\sum_{n=1}^{N} \left| e_{n}^{*} \right|^2
\]
(choose \( m \) if want padding/fit).

1) \( N_1 = M \), \( N_2 = n-1 \), no zero padding. (Covariance Method)

We were talking of Least squares.

\[
\begin{bmatrix}
y(m) \\
x(m) \\
\end{bmatrix}
\]
\( n = 0, \ldots, n-1 \)

\[
C(m) = y(m) - C^H x(m)
\]

we want \( \min_c \sum_{n=0}^{n-1} |C(m)|^2 \)

no assumptions of modeling distribution...
\[
\begin{bmatrix}
Y^X \\
y^X_{[0]} \\
y^X_{[1]} \\
y^X_{[N-2]} \\
y^X_{[N-1]}
\end{bmatrix} =
\begin{bmatrix}
X^Y \\
x^Y_{[0]} \\
x^Y_{[1]} \\
x^Y_{[N-2]} \\
x^Y_{[N-1]}
\end{bmatrix}c +
\begin{bmatrix}
e(c_0) \\
e(c_1) \\
e(c_2) \\
e(c_{N-2}) \\
e(c_{N-1})
\end{bmatrix}
\]

\[\hat{Y} = X + e\]

\[(X^*X)_{LS} = X^*y\quad \text{(the normal equations)}\]

if \(X^*X\) is invertible:

\[C_{LS} = (X^*X)^{-1}X^*y = X^+y\quad \text{(pseudo-inverse)}\]

\[y_{LS} = X C_{LS}\quad \text{(the approx of } y)\]

\(y_{LS}\) is the best projection of \(y\) onto the range space of \(X\).

\[y_{LS} = X C_{LS} = XX^+y = P_x y\]

\(P_x\) is the orthogonal projection matrix onto the range \(\text{range}(X)\).

\[P_x = P_{x^1}^1\]

\[C_{LS} = y - y_{LS} = (I - P_x)y = P_{x^1}^1y \perp R(x)\]

\(P_{x^1}^1\) is also the proj. mat.
Weighted Least Squares:

\[ J(w) = e^H W e \quad (W = W^H \text{ and positive definite}) \]

maybe some errors are more important than others. (before we had \( W = I \))

\[ J(w) = (y - x^T)^T W (y - x^T) \]  \( \min_c J(w) \) s.t. \( w \in S_r^+ \)

Solution:

\[
(x^T W x) \mathbf{w}_{LS} = x^T W y
\]

Time Series:

\( x[n], n = 0, \ldots, N-1 \)

we want Linear prediction 1-step:

\[
e_{n}[n] = x[n] + \sum_{k=3}^{m} a[k] x[n-k]
\]

Zero padding in past samples:

<table>
<thead>
<tr>
<th>( e^T \mathbf{x}[n] )</th>
<th>( x[0], 0, \ldots, 0 )</th>
<th>( [1] )</th>
</tr>
</thead>
<tbody>
<tr>
<td>( e^T \mathbf{x}[n-1] )</td>
<td>( \ldots, x[0], 0 )</td>
<td>( a )</td>
</tr>
<tr>
<td>( e^T \mathbf{x}[n-2] )</td>
<td>( \ldots, x[0], 0 )</td>
<td>( a )</td>
</tr>
<tr>
<td>( e^T \mathbf{x}[n-3] )</td>
<td>( \ldots, x[0], 0 )</td>
<td>( a )</td>
</tr>
<tr>
<td>( e^T \mathbf{x}[n-4] )</td>
<td>( \ldots, x[0], 0 )</td>
<td>( a )</td>
</tr>
</tbody>
</table>

Zero padding in future samples:

<table>
<thead>
<tr>
<th>( e^T \mathbf{x}[n+N] )</th>
<th>( 0, x[1], \ldots, x[n+1] )</th>
</tr>
</thead>
<tbody>
<tr>
<td>( e^T \mathbf{x}[n+N+1] )</td>
<td>( \ldots, 0, x[N-1] )</td>
</tr>
<tr>
<td>( e^T \mathbf{x}[n+N+2] )</td>
<td>( \ldots, 0, x[N-1] )</td>
</tr>
</tbody>
</table>

\( (m+1 \times d) \)
\[ \min_a \sum_{n=1}^{N_2} |e_m^n| \]

we have \(4 = 2 \times 2\) choices for \(N_1, N_2\):

1) \(N_1 = 0, N_2 = N-1\)  
   no zero padding  
   zeros at front

2) \(N_1 = M, N_2 = N-1\)  
   no zero padding  
   zeros at front  
   (covariance method)

3) \(N_1 = M, N_2 = N+M-1\)  
   zeros at back

4) \(N_1 = 0, N_2 = N+M-1\)  
   zeros both back & front

the most popular:  

2) \(\) (no padding at all)  
and  

4) \(\) (padding in both sides).

2) is called covariance method \(\) (forward only approach)

4) is called correlation method \(\) it leads to  

\[ X^H X \text{ Toeplitz} \rightarrow \text{easy for computation (LDA)} \]

We can get a similar equation system for backward prediction.

for covar method \(\) (no zeros padding); forward only:

\[ e_f = X \begin{bmatrix} a \end{bmatrix} \quad (X = \begin{bmatrix} x[t_1] & \cdots & x[t_g] \\ \vdots & \ddots & \vdots \\ x[t_{N-1}] & \cdots & x[t_{N-M-1}] \end{bmatrix}) \]

(first col is)

like the \( x \) in \( L_s \)

Let's do the same for backward prediction:
\[ c_m^n = x[n-m] + \sum_{k=0}^{n-m} b_k x[n-k] \]

and we know the relation \( a = J b^* \)

\[ c_m^b[n] = [x[n-m], x[n-m+1], \ldots, x[n]] \begin{bmatrix} 1 \\ b_{m-1}^* \\ \vdots \\ b_0^* \end{bmatrix} \]

notice \( b_{m-1} = a_1 \)
\( b_0 = a_0 \)

so:
\[ c_m^b[n] = [x[n-m], \ldots, x[n]] \begin{bmatrix} 1 \\ a \end{bmatrix} \]

now let's stack these equations for different m's (no zero padding "backward only")

\[
\begin{bmatrix}
  c_m^b[m] \\
  \vdots \\
  c_m^b[n-1]
\end{bmatrix} =
\begin{bmatrix}
  x[n], \ldots, x[n-m] \\
  \vdots \\
  x[n-1-m]
\end{bmatrix}
\begin{bmatrix}
  1 \\
  a
\end{bmatrix}
\]

\[ c_b = X^* J \begin{bmatrix} 1 \\ a \end{bmatrix} \] (flipping the column order of matrix \( X \) we had before)

so we have 2 equations:
\[
\begin{cases}
  c_p = X[a] \\
  c_b = X^*[a]
\end{cases}
\]
Let's add them together:

\[
\begin{bmatrix}
  e^a \\
e^b
\end{bmatrix} =
\begin{bmatrix}
x & \cdots & 1
\end{bmatrix}
\begin{bmatrix}
a
\end{bmatrix}
\]

we can solve for \( a \) a least squares solution in the Forward-Backward approach of LS for time series.

\[
\min_a \| e^a \|_2^2
\]

the \( FB \) approach has been found to be more reliable than fixed-only.

---

**Recursive Least Squares** (computational technique)

Two main ideas:

\( a) \) how to update \( a \) based on new samples (efficiently)

\( b) \) for non-stationary signals: de-emphasize older samples (this is a type of weighted Least Squares).

samples \( 0, \ldots, (n-1) \), \( Y[l], X[l] \) \( l = 0, \ldots, n-1 \).

\[
\begin{bmatrix}
y[0] \\
y[1] \\
\vdots \\
y[n-1]
\end{bmatrix} =
\begin{bmatrix}
x[0]^T \\
x[1]^T \\
\vdots \\
x[n-1]^T
\end{bmatrix}
\begin{bmatrix}
c + e_{n-1}
\end{bmatrix}
\]

\[
y[n] = X[n-1] c + e_{n-1}
\]
the LS solution will satisfy the normal equation:

\[(X^H \Phi \Phi X^H)^{-1} X^H \Phi \Phi y = X^H \Phi \Phi y_0\]

as \(n\) progresses, these vectors/matrices grow.
(though the final equation remains \(n \times 1\))

At time \(n\):

\[
\begin{bmatrix}
X^{n-1} \\
X^{n-2} \\
\vdots \\
X^n
\end{bmatrix}
= \begin{bmatrix}
C \\
C + E_n
\end{bmatrix}
\]

we can solve LS again for the bigger equation:

\[(X^H \Phi \Phi X^H)^{-1} \Phi \Phi y = X^H \Phi \Phi y_0\]

but we already done the work to get \(C_{n-1}\), how can we get \(C_n\) efficiently?

Let's introduce the weighting matrix \(W_{n+1}(\alpha)\) and we want to relate it to \(W_n\).

\[
W_{n+1}(\alpha) = \begin{bmatrix}
\alpha^n & 0 \\
\alpha^{n-1} & \alpha \\
0 & \alpha
\end{bmatrix}
\]

if \(n=1\) we get \(W_1 = I\).

if signal changes quickly we can select smaller \(n\), and the past samples will be forgotten quickly.
\[ W(n-1) = \begin{bmatrix} \mathbf{W}_0 & 0 \\ 0 & \mathbf{W}_1 \end{bmatrix} \quad W(n) = \begin{bmatrix} \mathbf{W}_0 & 0 \\ 0 & \mathbf{W}_1 \end{bmatrix} \]

\( \mathbf{r} \) - forgetting factor

weighted LS solution for \( n-1 \):

\[
\hat{\mathbf{d}}(n-1) = \begin{bmatrix} \mathbf{R}^{-1} \mathbf{W}(n-1) \mathbf{C}(n-1) \\ \mathbf{R}^{-1} \mathbf{W}(n-1) \mathbf{x}(n-1) \end{bmatrix}
\]

and for time \( n \):

\[
\hat{\mathbf{d}}(n) = \begin{bmatrix} \mathbf{R}^{-1} \mathbf{W}(n) \mathbf{C}(n) \\ \mathbf{R}^{-1} \mathbf{W}(n) \mathbf{x}(n) \end{bmatrix}
\]

\( \hat{\mathbf{d}} \) estimates of data correlation and the cross-correlation

\[
\mathbf{C}(n-1) = \mathbf{R}(n-1) \mathbf{d}(n-1)
\]

\[
\mathbf{C}(n) = \mathbf{R}(n) \mathbf{d}(n)
\]

\[
\hat{\mathbf{d}}(n) = \begin{bmatrix} \mathbf{R}^{-1} \mathbf{W}(n) \mathbf{C}(n) \\ \mathbf{R}^{-1} \mathbf{W}(n) \mathbf{x}(n) \end{bmatrix} = \begin{bmatrix} \mathbf{R}^{-1} \mathbf{W}(n) \mathbf{x}(n) + \mathbf{x}(n)^T \mathbf{R}^{-1} \mathbf{y}(n) \end{bmatrix}
\]

\[
\hat{\mathbf{R}}(n) = \mathbf{R}^{-1} \mathbf{W}(n) \mathbf{x}(n) = \begin{bmatrix} \mathbf{R}^{-1} \mathbf{W}(n) \mathbf{x}(n) + \mathbf{x}(n)^T \mathbf{R}^{-1} \mathbf{y}(n) \end{bmatrix}
\]

Can we also efficiently update \( \hat{\mathbf{R}}(n) \)?

we use the matrix inversion lemma.
Matrix Inversion Lemma:

\[(A + BCD)^{-1} = A^{-1} - A^{-1}B(C^{-1} + DA^{-1}B)^{-1}DA^{-1}\]

\[A, B, C, D : m \times n, n \times r, r \times r, r \times m\]

we need A and C to be invertible.

to verify this lemma is true - simply multiply with \(A + BCD\) and get I.

in our case: \(A = 2^R \tilde{G} - I\) and we already have access to precomputed \(A^{-1}\), \(B = XGJ\), \(C - I\), \(D = X^{H}GJ\)

Back to RLS

\[
\begin{bmatrix}
  e^x_{t+3} \\
  \vdots \\
  e^x_{t+n-3}
\end{bmatrix}
= 
\begin{bmatrix}
  y^x_{t+3} \\
  \vdots \\
  y^x_{t+n-3}
\end{bmatrix}
- 
\begin{bmatrix}
  x^y_{t+3} \\
  \vdots \\
  x^y_{t+n-3}
\end{bmatrix} C
\]

\[
y_{t+n-3} = X(n-3)_{(n \times m)}
\]

\[
x_{t+n-3}^{H}W_{t+n-3}^{H} = x_{t+n-3}^{H}W_{t+n-3}^{H} + y_{t+n-2}
\]

\[
\hat{R}_{t+n-1} \hat{C} - \hat{I} = \hat{A}_{t+n-1}^{-1}
\]
\[ y_n = \begin{bmatrix} y_{n-1} \\ y_{n-1} \\ \vdots \\ y_1 \\ 1 \end{bmatrix}, \quad X(n) = \begin{bmatrix} X(n-1) \\ X(n-1) \\ \vdots \\ X(1) \\ x_0 \end{bmatrix} \]

\[(n+1)x_n \quad \ast \quad (n+1)x_m \]

\[ x^T_{n+1} c_{n+1} = x^T_n c_n + y_n \quad (R(n) c_n = 1) \]

we chose to use:

\[ W_{n} = \begin{bmatrix} \alpha^n & 0 \\ 0 & 1 \end{bmatrix} = \begin{bmatrix} \alpha^{m-n} & 0 \\ 0 & 1 \end{bmatrix} \]

\[ z(n+1) = \alpha^n z_n + x(n) y^* + y(n) \]

\[ R(n) = \alpha R(n-1) + x(n) x^* + y(n) \]

we want to also get a fast update to \( R^{-1} \).

Matrix Inversion Lemma:

\[ (A + BCD) = A - A^{-1} \frac{B(C^{-1} + D^* B^{-1} A^{-1})}{1 + b^* A^{-1} b} \]

in our case \( C \) is a scalar so inverting it is very easy. also \( B = B^* \).

\[ (A + b^* b^{-1}) = A - A^{-1} \frac{b b^* A^{-1} b}{1 + b^* A^{-1} b} \]

looking ahead lets also check what's \( (A + b b^*)^{-1} b \)

\[ = A^{-1} b + \frac{a b^* b A^{-1} b}{1 + b^* A^{-1} b} = (1 - b^* A^{-1} b) A^{-1} b \frac{1}{1 + b^* A^{-1} b} \]
In our case:

\[ R_{m+1} = \frac{R_{m} - \alpha_{m} \cdot x_{m} \cdot y_{m} \cdot \hat{R}_{m}^{T} \cdot \hat{R}_{m}^{-1} \cdot \hat{R}_{m}^{-1}}{1 + \frac{\alpha_{m} \cdot x_{m} \cdot y_{m} \cdot \hat{R}_{m}^{T} \cdot \hat{R}_{m}^{-1} \cdot \hat{R}_{m}^{-1}}{\hat{R}_{m}^{T} \cdot \hat{R}_{m}^{-1} \cdot R_{m}}} \]

Simplify:

\[ g_{m} = 2^{a_{m}} \cdot R_{m} \cdot x_{m} \cdot y_{m} \]

\[ g_{m} = \frac{\hat{R}_{m}^{T} \cdot \hat{R}_{m}^{-1} \cdot \hat{R}_{m}^{-1} \cdot \hat{R}_{m}^{T} \cdot \hat{R}_{m}^{-1} \cdot R_{m}}{1 + \frac{\alpha_{m} \cdot x_{m} \cdot y_{m} \cdot \hat{R}_{m}^{T} \cdot \hat{R}_{m}^{-1} \cdot \hat{R}_{m}^{-1}}{\hat{R}_{m}^{T} \cdot \hat{R}_{m}^{-1} \cdot R_{m}}} \]

So:

\[ \hat{R}_{m}^{T} \cdot \hat{R}_{m}^{-1} \cdot y_{m} \cdot g_{m} \]

So if we have already \( \hat{R}_{m}, R_{m}, c_{m} \), we get new sample \( x_{m}, y_{m} \) we compute \( \hat{R}_{m}^{T} \cdot \hat{R}_{m}^{-1} \) and then \( c_{m} \).

Update for \( \hat{R}_{m}^{T} \cdot \hat{R}_{m}^{-1} \) takes \( O(n^2) \) computation.

Updating \( c_{m} \):

\[ \hat{R}_{m}^{T} \cdot c_{m} \cdot x_{m} = \hat{R}_{m} \cdot x_{m} \cdot y_{m} \]

\[ (\hat{R}_{m}^{T} \cdot x_{m} \cdot y_{m}) \cdot c_{m} \cdot x_{m} = \hat{R}_{m} \cdot x_{m} \cdot y_{m} \]

\[ \hat{R}_{m} = \hat{R}_{m} \cdot c_{m} \cdot x_{m} \cdot y_{m} \]

\[ \hat{R}_{m} = \hat{R}_{m} \cdot c_{m} \cdot x_{m} \cdot y_{m} \]

\[ \hat{R}_{m} = \hat{R}_{m} \cdot c_{m} \cdot x_{m} \cdot y_{m} \]

Mult by \( \hat{R}_{m}^{T} \) getting:

\[ c_{m} = c_{m} + \frac{\alpha_{m} \cdot x_{m} \cdot y_{m} \cdot \hat{R}_{m}^{T} \cdot \hat{R}_{m}^{-1} \cdot \hat{R}_{m}^{-1}}{1 + \frac{\alpha_{m} \cdot x_{m} \cdot y_{m} \cdot \hat{R}_{m}^{T} \cdot \hat{R}_{m}^{-1} \cdot \hat{R}_{m}^{-1}}{\hat{R}_{m}^{T} \cdot \hat{R}_{m}^{-1} \cdot R_{m}}} \cdot c_{m} \]
(c_{H} - x) + g[n] e^{*}[n] 

So, our recipe:

\[ R[n-1] \rightarrow P[n-1] \rightarrow g[n] \rightarrow P[n], \text{ and } c[n] \]

RLS update takes \( O(m^2) \)

(LMS update was \( O(m) \))

for initializing:

1) we can wait till \( n = m \)

compute a full LMS operation (including inverting a matrix) just once, and from then on = RLS.

2) initialize \( C[0] = 0 \_m \)

initialize \( P[0] = \frac{1}{\delta} \_I \_m \) (\( \delta \) small).

since the system forgets quickly (a small) this will work well.

\[ R[n] = \sum_{k=0}^{n} x[k]x[k]^H + \frac{\delta}{\delta} \_I \]

for tracking \( n \) close to 1 if things are close to stationary. \( \frac{\delta}{\delta} \) small \( \rightarrow \) for fast forgetting. \( \frac{\delta}{\delta} \) for tracking quick changes but in practice \( \frac{\delta}{\delta} \approx 0.5 \)
is too small (not integrating enough samples) good area for \( x : 0.9 \leq x \leq 1 \).

\[ y(t) = c_0 g(t) + e_0(b + e_m) \quad \text{(model we want to)} \]
\[ c_0(t) = c_0(t-1) + \nu \Delta s \quad \text{(track)} \]
\[ e_0(t) \sim N(0, \sigma_e^2) \quad \nu \sim N(0, \sigma^2) \]

(for LMS we get:
\[ P_{x(t)} = \frac{1}{\Gamma} \Rightarrow mc_1 + \frac{c_2}{\mu} \]

for RLS we get:
\[ P_{x[t]} = \frac{M(1-\alpha)}{2} \sigma_e^2 + \frac{1}{2(1-\alpha)} \text{tr}(R) \]

\( R = E(x(t)x(t)^T) \)

-- again: RLS is just a computational tool
\( \alpha = 1: \text{RLS does LS}, \alpha < 1: \text{RLS does WLS} \)

---

Final exam: Tue 11/20.
2 sheets of notes hand written.

emphasis on material past the midterm:
1. Lattice filters (\( \hat{c} \), \( \hat{a} \))
2. LMS (conv. mean, conv. tracking)
3. LS, RLS. (LS, RLS, orthogonality = projection matrix)

---

Bring paper & calculator. Toeplitz = no inversion, use LDA.
HW 7. 9.3

\[ y = Xc + e \]
\[ C_{WS} = X^+y \]
\[ x^+ = (X^T X)^{-1}X^T \]

\[ P_x = XX^+ \quad \text{orth proj to } \mathbb{R}(X) \]

\[ E^0: \begin{bmatrix} e_1^T \, m_1 \\ \vdots \\ e_n^T \, m_n \end{bmatrix} = \begin{bmatrix} x^T(m_1) \cdots x^T(m_n) \\ a \end{bmatrix} \]

Power spec:

\[ P_m = \frac{1}{A_m(a)} \]

\[ P_m = \frac{\|e\|^2}{\text{num samples}} \]

\[ A_m(a) = 1 + \sum_{k=1}^{a} a_k \frac{a^k}{k} \]