STANFORD

Lecture 16

FIR Equalizer Design & Software

March 5, 2024

JOHN M. CIOFFI

Hitachi Professor Emeritus (recalled) of Engineering

Instructor EE379A – Winter 2024

Announcements & Agenda

- Announcements
- Final
  - Distribute end of class Thursday March 14
  - Due 25 hours later – designed for in-class
    - Few hours, but want to avoid time pressure
- PS8 Can read ahead and do PS8.4-5 (3.59 and 6.2 early)
  - Or just use the solutions to study and leave them out of PS8
  - Grader will adjust score accordingly on any PS8's to 3-5 problems (those attempted).
- Feedback (PS6)
  - 9-14 hours
  - $SNR_{MFB} = \bar{\mathcal{E}}_x \cdot \|h\|^2 / \sigma^2$ is
    - upper bound on best **receiver** SNR
  - $SNR = \bar{\mathcal{E}}_x / \sigma^2$ is at **transmitter.**
  - $\gamma_{MMSE-LE} = {}^{SNR_{MMSE-LE,U}} / {}_{SNR_{MFB}}$
    - Loss w.r.t. best possible receiver SNR
  - $\|h\|^2 = \|p\|^2$ if you see a p anywhere.
- PS7 – Pos-real R(D) factors if PWC satisfied.
  - See Appendix D for Proof.

- Optional Problem Set 8 = PS8 due Tuesday March 12 (no late)
  1. 3.14 Tomlinson Precoding
  2. 3.31 Diversity Channel
  3. 3.53 DFE Color Program
  4. 3.59 Multiband Optimization
  5. 6.2 2nd-Order PLL

- Today
  - Finish FIR linear
  - FIR DFE
  - dfecolor.m program
    - Examples
  - Adaptive Equalizers and channel identification

FIR ZFE

```
H=[.9 1 0 0
0 .9 1 0
0 0 .9 1];
>> wzfe=H(:,4)'*inv(H*H') =   0.2702   -0.5434   0.8227

>> MMSEzfisi=1-wzfe*H(:,4) =  0.1773
```

- Design applies MMSE-LE with infinite SNR.
  - Finite taps cannot guarantee zero ISI, so there is:

$$\sigma^2_{ZFE-ISI} = \bar{\mathcal{E}}_x - \boldsymbol{w} \cdot R_{Yx}$$

- The FIR ZFE can still be biased – look at position of $x_{k-\Delta}$ so $\boldsymbol{w} \cdot \boldsymbol{H}$ in position $\Delta + 1$

chan=wzfe*H =   0.2432   -0.2189   0.1970   0.8227

  - Bias removal thus inverts, so $\omega_{FIR-ZFE} = 1 \,/\, \boldsymbol{w} \cdot \boldsymbol{H} \cdot \mathbf{1}_\Delta$; this means residual ISI and noise increase by this factor also.

- Analysis must also add the scale the enhanced noise $\sigma^2_{ZFE} = \boldsymbol{w} \cdot R_{\boldsymbol{nn}} \cdot \boldsymbol{w}^*$.
- Total is $\sigma^2_{FIR-ZFE} = \omega_{FIR-ZFE} \cdot (\bar{\mathcal{E}}_x - \boldsymbol{w} \cdot R_{Yx} + \boldsymbol{w} \cdot R_{\boldsymbol{nn}} \cdot \boldsymbol{w}^*)$

```
>> chan = wzfe*H % =   0.2432   -0.2189   0.1970   0.8227
>> wnobias=1/chan(4)  %=   1.2155
>> SNRzfu=1/MMSEzfisi -1  % =   4.6402
>> (SNRzfu+1)/SNRzfu =   1.2155 % checks bias removal
>> enoise=.181*norm(wzfe)^2  =   0.1892
>> SNRall=10*log10(1/(wnobias*(MMSEzfisi+enoise)))  = 3.512  dB (< 3.78 dB for MMSE-LE,U)
```

FIR DFE

Section 3.7.4

Extend FIR to DFE

- Extend MMSE FIR criterion to feedback $e_k = x_{k-\Delta} - \boldsymbol{w} \cdot \boldsymbol{Y}_k + \boldsymbol{b} \cdot \boldsymbol{x}_{k-\Delta-1}$; $\sigma^2_{MMSE-LE} = \mathbb{E}[|e_k|^2]$,
  - where $\boldsymbol{b} = [b_1 \quad \cdots \quad b_{N_b}]$ and
- Define $\tilde{\boldsymbol{w}} \triangleq [\boldsymbol{w} \quad \boldsymbol{b}]$ and $\tilde{\boldsymbol{Y}}_k \triangleq \begin{bmatrix} Y_k \\ \boldsymbol{x}_{k-\Delta-1} \end{bmatrix}$ and then $\sigma^2_{MMSE-LE} = \mathbb{E}\left\{\left|x_{k-\Delta} - \tilde{\boldsymbol{w}} \cdot \tilde{\boldsymbol{Y}}_k\right|^2\right\}$.

where J_Δ is an $(N_f+\nu) \times N_b$ matrix of 0's and 1's, which has the upper $\Delta+1$ rows zeroed and an identity matrix of dimension $\min(N_b, N_f+\nu-\Delta-1)$ with zeros to the right (when $N_f+\nu-\Delta-1 < N_b$), zeros below (when $N_f+\nu-\Delta-1 > N_b$), or no zeros to the right or below exactly fitting in the bottom of J_Δ (when $N_f+\nu-\Delta-1 = N_b$).

- Auto and cross correlation are:
  - $R_{\tilde{\boldsymbol{Y}}\tilde{\boldsymbol{Y}}} \triangleq \begin{bmatrix} R_{\boldsymbol{YY}} & \bar{\mathcal{E}}_x \cdot \boldsymbol{H} \cdot J_{\boldsymbol{\Delta}} \\ \bar{\mathcal{E}}_x \cdot J^*_\Delta \cdot \boldsymbol{H}^* & \bar{\mathcal{E}}_x \cdot I_{N_b} \end{bmatrix}$ and $R_{\tilde{\boldsymbol{Y}}\boldsymbol{x}} \triangleq \begin{bmatrix} \bar{\mathcal{E}}_x \cdot \boldsymbol{H} \cdot \boldsymbol{1}_{\boldsymbol{\Delta}} \\ 0 \end{bmatrix}$.

Detailed algebra in Sec 3.7.4 Including some interesting interpretations (see EE379B)

- The solution is
  - $\tilde{\boldsymbol{w}} = R_{x\tilde{\boldsymbol{Y}}} \cdot R^{-\boldsymbol{1}}_{\tilde{\boldsymbol{Y}}\tilde{\boldsymbol{Y}}}$; $\boldsymbol{w} = \boldsymbol{H}^*_\Delta \cdot \left(\boldsymbol{H} \cdot \boldsymbol{H}^* - \boldsymbol{H} \cdot \boldsymbol{J}_{\boldsymbol{\Delta}} \cdot \boldsymbol{J}^*_\Delta \cdot \boldsymbol{H}^* + \frac{1}{\bar{\mathcal{E}}_x} \cdot R_{NN}\right)^{-1}$; $\boldsymbol{b} = \boldsymbol{w} \cdot \boldsymbol{H} \cdot \boldsymbol{J}_{\boldsymbol{\Delta}}$,
  - $\sigma^2_{MMSE-DFE} = \bar{\mathcal{E}}_x - \boldsymbol{w} \cdot R_{\boldsymbol{Y}\boldsymbol{x}}$; $SNR_{MMSE-DFE,U} = \dfrac{\tilde{\boldsymbol{w}} \cdot R_{\tilde{\boldsymbol{Y}}\boldsymbol{x}}}{\bar{\mathcal{E}}_x - \tilde{\boldsymbol{w}} \cdot R_{\tilde{\boldsymbol{Y}}\boldsymbol{x}}}$; $\gamma_{\,MMSE-DFE,U} = \dfrac{SNR_{MMSE-DFE,U}}{SNR_{MFB}}$.

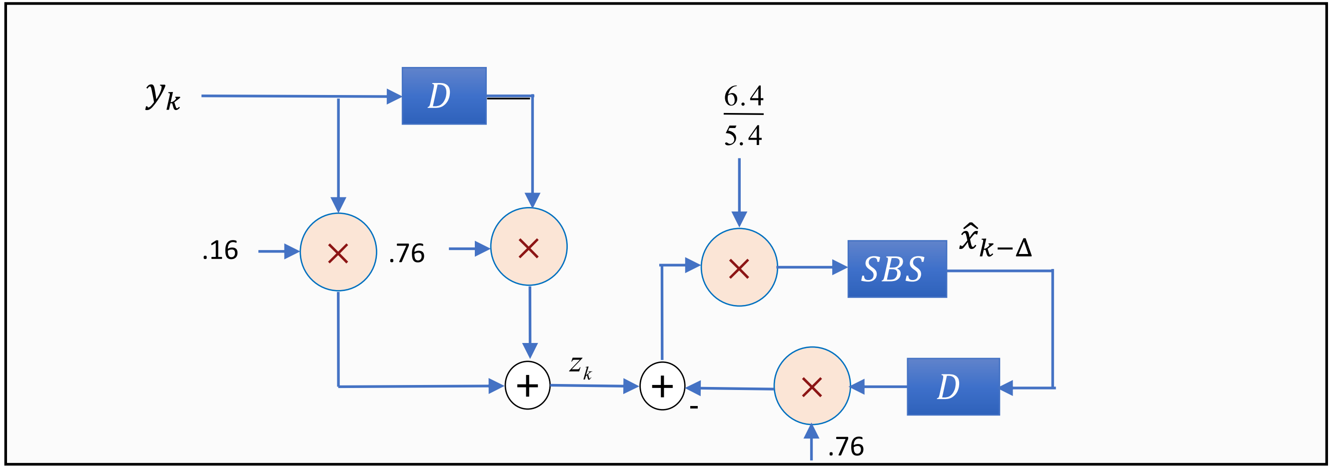
Return to 1+.9D^{-1} example

- $N_f = 2\ ;\ \bar{\mathcal{E}}_x = 1\ ;\ N_b = 1\ ;\ \Delta = 1\ ;\ \sigma^2 = .181$

```
>> H=
   0.9000    1.0000         0
        0    0.9000    1.0000

>> Jdel=[0 ; 0 ; 1];  % Nf x nu with top delta+1 rows zeroed
>> onedel=[0 ; 1 ; 0];

>> wdfe=onedel'*H'*inv(H*H'+.181*eye(2)-H*Jdel*Jdel'*H') =
   0.1556    0.7668
>> b=wdfe*H*Jdel =    0.7668


>> sigdfe=1-wdfe*H*onedel =    0.1542
>> SNRdfeu=10*log10(1/sigdfe - 1) =    7.3911   (Wow!)


>> >> gammadfe=10-SNRdfeu =    2.6 dB
```

Now we're cooking!

Block Diagram of MMSE-DFE Example



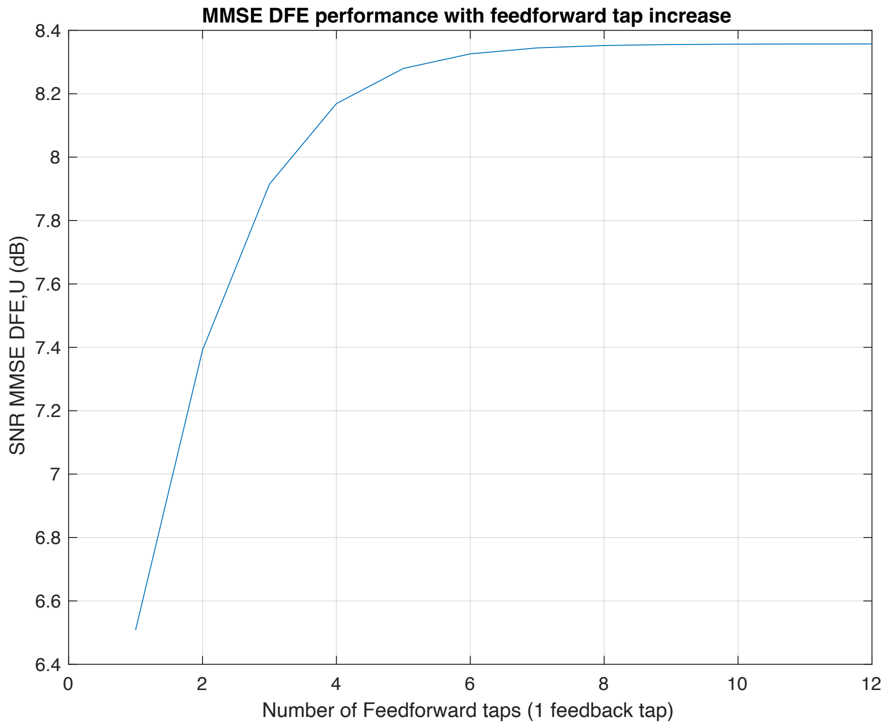
- Feedforward section is “noncausal” (with delay largest tap is last), “maximum phase.”
  - This is consistent with infinite-length being anticausal.
  - Implementation absorbs the bias-removal scaling into both taps and save one multiply

- Feedback is $B(D) = 1 + .76 \cdot D$ causal, and not ZF’s .9 but already close to infinite length unbiased .725.

- Unbiasing incorporation is:

```
[wdfe b]*(SNRdfeu+1)/SNRdfeu %=   0.1767   0.8706   0.8706
```

Increase number of forward taps

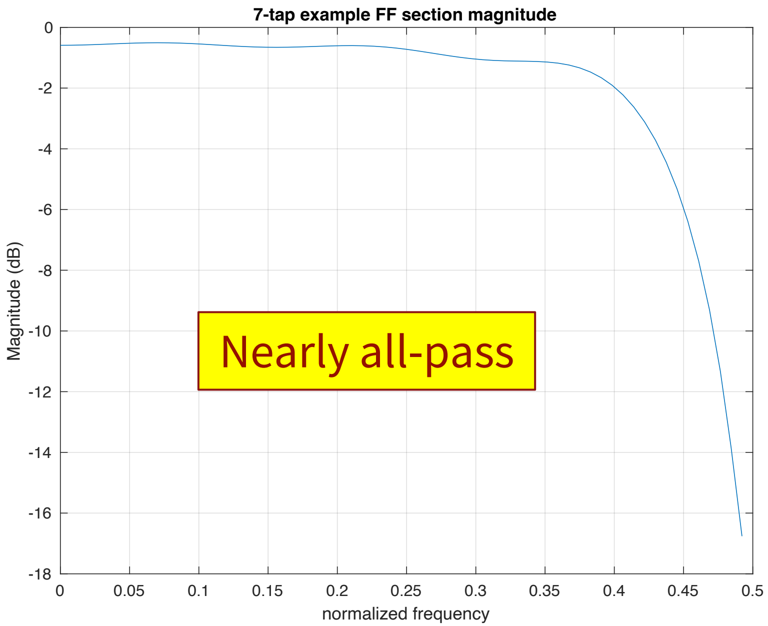


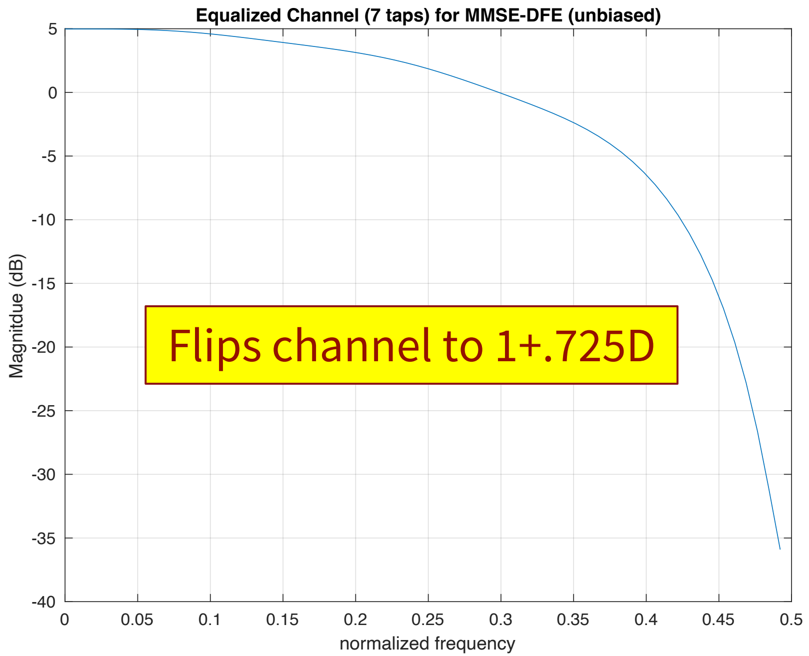
8.4dB, fairly low complexity

There is still more yet – Anyone guess where?

- This is typical of DFE on channels with severe ISI – much better than LE.
  - We will deal shortly with issue of "suppose the decision is wrong" (error propagation).

Eliminates Noise Enhancement





- Noise enhancement no longer occurs.
- FF is almost all pass, except for MMSE trade of noise reduction with residual ISI.

dfecolor.m

Section 3.7.6

The program – many 379 students

- This program created, and heavily used, by former students – many cases beyond class.

```
>> help dfecolor
  --------------------------------------------------------------
  [dfseSNR,w_t] = dfecolor(l,p,nff,nbb,delay,Ex,noise);

  INPUTS
  l     = oversampling factor
  h     = pulse response, oversampled at l (size)
  nff   = number of feed-forward taps
  nbb   = number of feedback taps
  delay = delay of system <= nff+length of p - 2 - nbb
        if delay = -1, then this program chooses best delay
  Ex    = average energy of signals
  noise = noise autocorrelation vector (size l*nff)
    so white noise is [1 zeros(l*nff-1)]
  NOTE: noise is assumed to be stationary

  OUTPUTS
  dfseSNR = equalizer SNR, unbiased in dB
  w_t = equalizer coefficients [w -b]
  opt_delay = optimal delay found if delay =-1 option used.
          otherwise, returns delay value passed to function
  created 4/96;
  --------------------------------------------------------------
```

The program allows “colored” noise (as any noise whitening in practice absorbes into the equalizer)

Return to $H(D) = 1 + .9 \cdot D^{-1}$

- The command is $l = 1$; $h = [.9 \;\; 1]$,; $N_f = 3$; $N_b = 0$; $\Delta = -1$; $\bar{\mathcal{E}}_x$=1 ; noise= .181 & white

```
>> [SNRle, wle, msdelay ]=dfecolor(1,[.9 1],3,0,-1,1,.181*[1 zeros(1,2)])

SNRle =    3.7979 dB
wle =  -0.2277    0.5038    0.2243
msdelay =    2
```

- For the MMSE-LE with 7 taps:

```
[SNRle, wle, msdelay ]=dfecolor(1,[.9 1],7,0,-1,1,.181*[1 zeros(1,6)])

SNRle =    5.3956 dB
wle =  -0.0789    0.1745   -0.3072    0.5050    0.3011   -0.1710    0.0773
msdelay =    4
```

- For Slide 11's SNR graph:

```
>> for nff=1:20   [dfseSNR(nff),~]=dfecolor(l,h,nff,nbb,-1,Ex,.181*[1 zeros(1,nff-1)]));  end
>> plot(dfseSNR)
```

- For the 3-tap ZFE:

```
>> [SNRzfu,w ]=dfecolor(1,[.9 1],3,0,-1,1,0*[1 zeros(1,2)])
>> chan=conv(wzfe,[.9 1]) =    0.2432   -0.2189    0.1970    0.8227
>> SNRzfu=10^(SNRzfu/10) % =    4.6404
>> wnobias = (SNRzfu+1)/SNRzfu =    1.2155
>> MMSEzf=1/(SNRzfu+1) =    0.1773
>> enoise=.181*norm(wzfe)^2  =    0.1892
>> SNRall=10*log10(1/(wnobias*(MMSEzf+enoise)))   = 3.5120  dB
```

For the DFE with various number of taps

- 3-tap MMSE-DFE

```
>> [SNRdfeu, wdfe, msdelay ]=dfecolor(1,[.9 1],2,1,-1,1,.181*[1 zeros(1,1)])

SNRdfeu =   7.3911 dB
wdfe =   0.1556   0.7668  -0.7668 this is the feedback filter  green is feedforward filter
msdelay =    1
```

- 7-tap MMSE-DFE

```
>> [SNRdfeu, wdfe, msdelay ]=dfecolor(1,[.9 1],6,1,-1,1,.181*[1 zeros(1,5)])

SNRdfeu =   8.3259
wdfe =   0.0290  -0.0642   0.1131  -0.1859   0.2982   0.6374  -0.6374
msdelay =    5
```

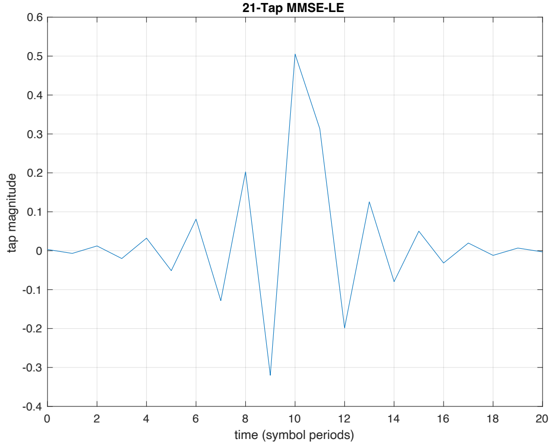
- 3-tap ZF-DFE

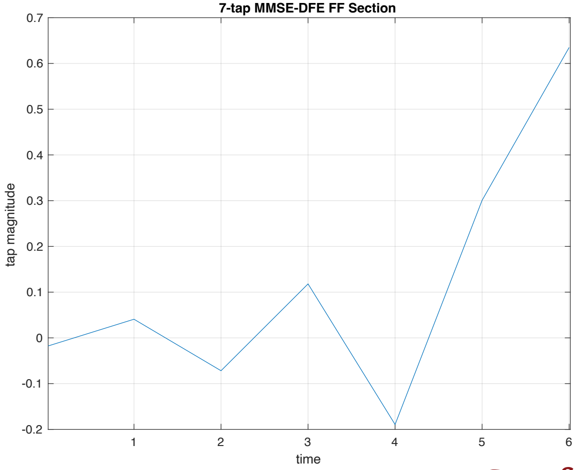
```
>> [SNRzdfeu, wzdfe, msdelay ]=dfecolor(1,[.9 1],2,1,-1,1,0*[1 zeros(1,0)])
SNRzdfeu =  159.5459 dB
wzdfe =    0   1.1111   -1.1111  (10/9)
msdelay =    1
>> .181*(10/9)^2 =    0.2235
>> 10*log10(1/ans) =    6.5081 dB < 7.4 dB < 8.3 dB
```

ZF-DFE will need many feedforward taps to convert the channel to minimum phase and then work properly, while MMSE-DFE sees such conversion as insignificant anyway w.r.t. noise.

Some other related uses

```
21 taps, LE?
>> [SNRlsu, wlse21, msdelay ]=dfecolor(1,[.9 1],21,0,-1,1,.181*[1
zeros(1,20)])

SNRlsu =    5.6830 dB
msdelay =    11
>>plot(wlse21)

[snrdfeu,wdfe7, del]=dfecolor(1,[.9 1],7,1,-1,1,.181*[1 zeros(1,6)])

snrdfeu =    8.3447

wdfe7 =
  -0.0184   0.0408  -0.0718   0.1180  -0.1893   0.3008   0.6350  -0.6350
del =     6

>> plot(wdfe7(1:7))
```

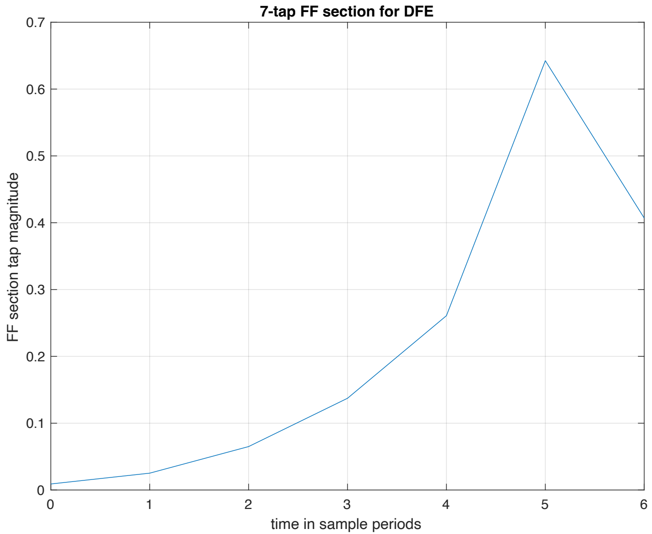


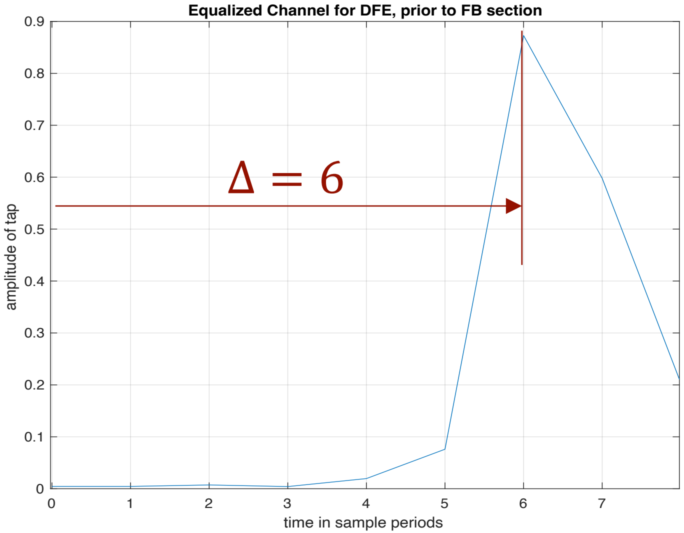


- Note LE delay = 10+1 (center tap + nu positions) anticaus chan
  - Need to count from time zero for DFE, this is max phase.
- DFE delay is all FF section as anticausal (7-1 = 6).

Complex Example?

```
>> [snrdfeu,wdfe7]=dfecolor(1,[-1/2 (1+i/4) -i/2],7,2,-1,1,.15625*[1 zeros(1,6)])
snrleu =    8.3651
wdfe7 =
0.0088 + 0.0019i   0.0248 + 0.0046i   0.0637 + 0.0128i   0.1319 + 0.0382i   0.2578 + 0.0395i   0.6417 - 0.0315i  -0.4070 + 0.0000i
0.4227 + 0.4226i  -0.0000 - 0.2035i
Delta =6
plot([0:6], abs(wdfe7(1:7)))               ; plot([0:8],abs(conv(wdfe7(1:7),[-1/2 (1+i/4) -i/2])))
```

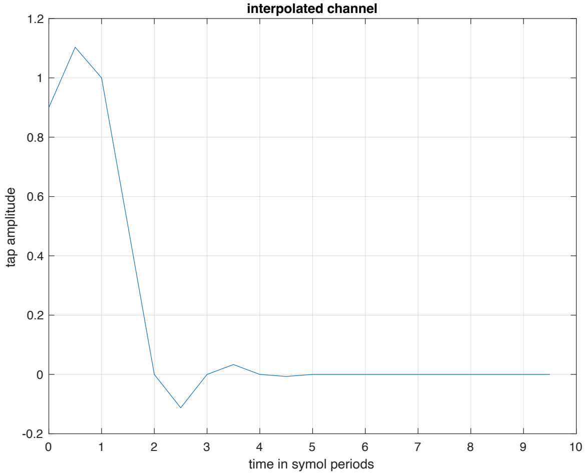


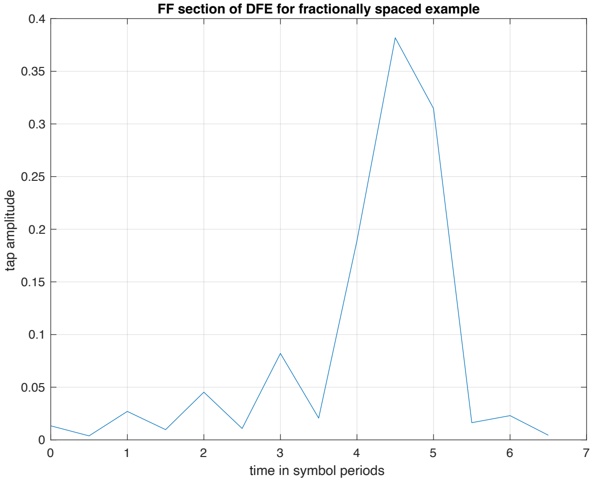


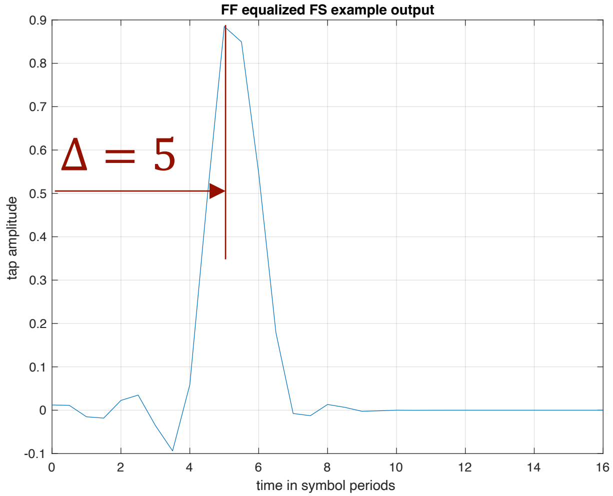
FSE Example for $1+.9D^{-1}$ interpolated

- Interpolate 1+.9D^{-1} channel to 2x (no longer "sharp cut off" at Nyquist)

```
newchan=interp([.9 1 zeros(1,8)],2); 0.5*norm(newchan)^2 % = 1.6461 < 1.81  % sets receiver input SNR the same.
>> plot(newchan)

>> [snrnewchan,wdfenewchan,delnewchan]=dfecolor(2,newchan,7,1,-1,1,.1646*[2 zeros(1,13)])
snrnewchan =   8.8578
wdfenewchan =  0.0133  -0.0038  -0.0270   0.0097   0.0453  -0.0108  -0.0821  -0.0206   0.1891   0.3818   0.3147   0.0163   0.0230   0.0044  -0.5465
delnewchan =    5
```







Why isn't fb tap = .633 (or .73 if unbiased)? What is this .55?
It's actually outperforming earlier infinite-length MMSE-DFE, why?

Error Propagation

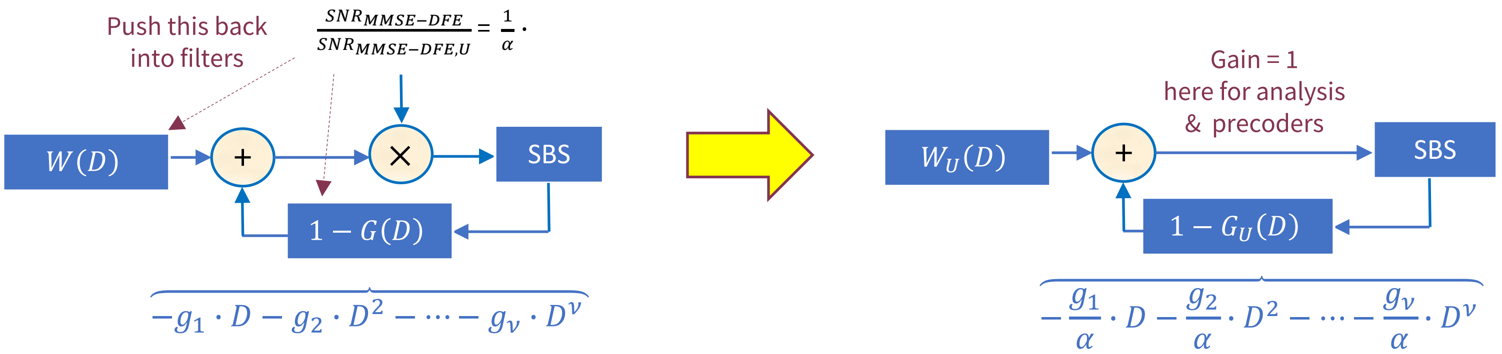
Section 3.7.7

Error Propagation

- If one error is made, then another is likely because of its feedback.
- This **error propagation** can seriously reduce performance if not addressed properly.
- With large coded constellations with small individual subsymbol's $d_{min,ss}$ (not the overall code d_{min}):
  - Error propagation has higher probability (so may require a "list" of feedback outputs).
  - FDTS problem on PS7.

There are many good solutions that eliminate or reduce error prop, but each can introduce additional issues.

Reminder/Refresher



Example

- $1+.9\cdot D^{-1}$ channel has $G_U(D)$=$1+.7D$, so
  - $y_k = x_k + .7\cdot x_{k-1}$ with zero noise,
  - $z_{U,k} = y_k - .7\cdot \hat{x}_{k-1}$.

k	-1	0	1	2	3
x_k	-1	+1	-1	+1	+1
y_k	-	0.3	-0.3	0.3	1.7
$z_{U,k}$	-	-0.4	0.4	-0.4	2.4
$\hat{x}_k$	+1	-1	+1	-1	+1

- One error produces 3 more in this case:
  - 1 error → 2 errors with prob ½ ,
  - 1 error → 3 errors with prob ¼ ,
  - 1 error → 4 errors with prob $\frac{1}{8}$, &
  - Infinite burst has probability 0.

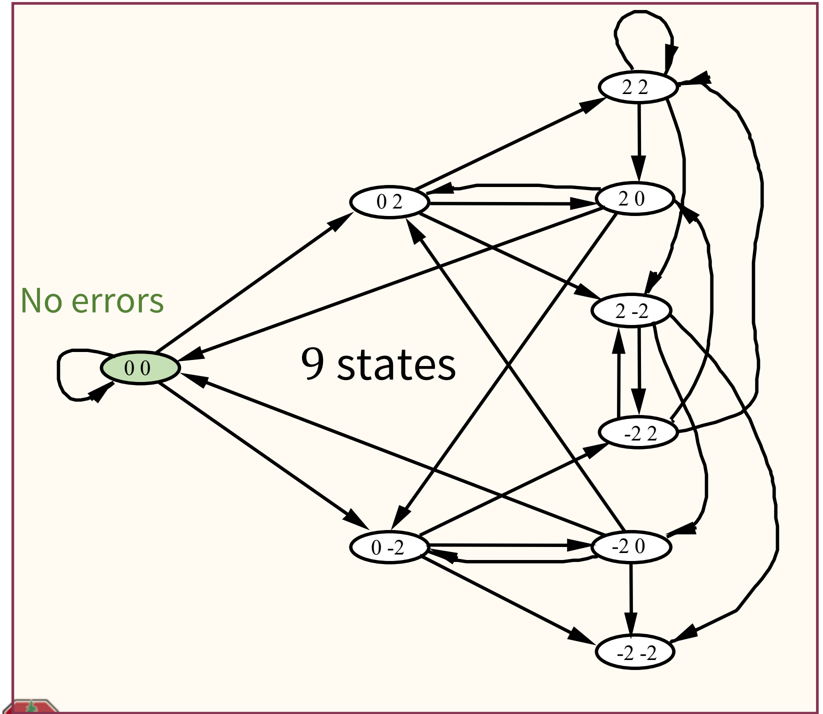
$$N_e = \sum_{i=1}^{\infty} i\cdot\left(\tfrac{1}{2}\right)^{i-1} = 2$$

Ave N_e only doubled, but nasty burst can occur.

State-Machine Error Model

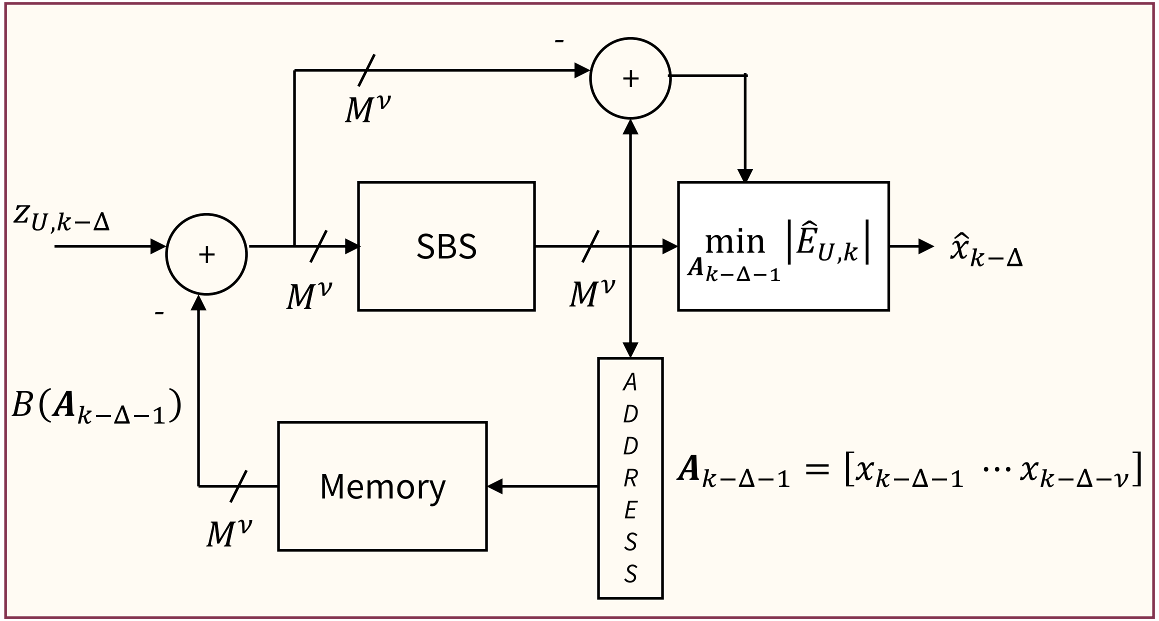
- It is possible to compute (with possibly enormous analysis calculations) the error-propagation-increased P_e.

$M = 2$ with $N_{fb} = 2$

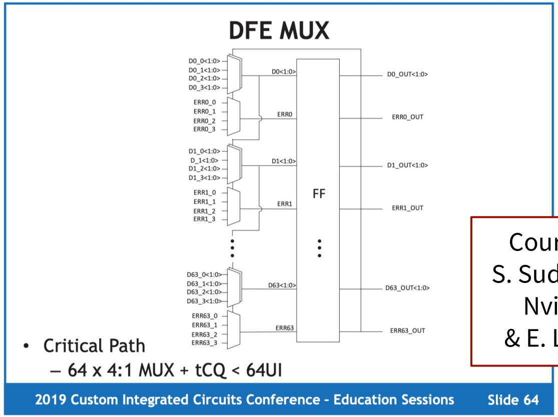


- Error events: $\boldsymbol{\epsilon}_{k-1} = [x_{k-1} - \hat{x}_{k-1} \quad \cdots \quad x_{k-N_b} - \hat{x}_{k-N_b}]$
- For $M = 2$, each error-event element can be ± 2 or 0.
- $\exists\ (2M-1)^{N_b}$ possible states; each has d_{min} offset in P_e calculation.
- Analysis finds the state machine's Markov stationary probability distribution (largest eigenvalue's vector for transition-probability matrix, see Appendix A).
- Then, a Markov-stationary-prob weighted sum of all states' P_e's leads to an exact overall P_e calculation.
- It can take much calculation – see Section 3.7.7.
- Designers prefer to avoid error propagation (and not then worry about computing it, nor incurring degradation).

Look ahead (general finite-delay tree search)



Look a bit complex?
It is, but sometimes done. There are other easier solutions though, although each has its own issue.



Courtesy
S. Sudhakar
Nvidia
& E. Liang

- FDTS essentially computes M^{ν} FB section outputs and checks error magnitude on all. It uses the smallest.
  - FDTS ensures (finite ν) that $SNR_{MMsE-DFE,U}$ is attained (no error propagation loss), but complex.
- This is an intermediate step towards Chapter 7's sequence detection for ISI (Section 7.2).
- FTDS computes a DFE soft metric for use in iterative decoding as extrinsic information from DFE.

Use packets with guard period.

Live Data	Dum my	Live Data	Dum my

- Data stream contains large packets with a few dummy symbols at end.
  - Rate loss is few percent or less.

- Any error propagation stops at packet end.

- If packet length is shorter than P_e^{-1} , then erred packets occur infrequently.
  - Outer code then (maybe with interleaving) obtains final desired performance.

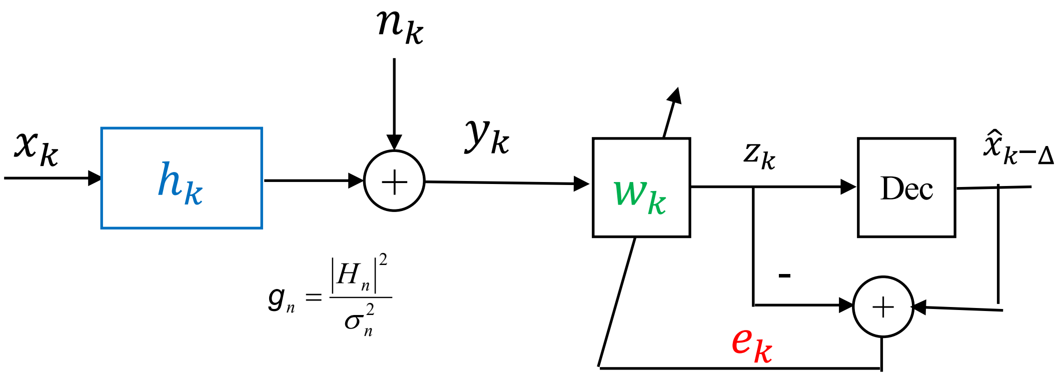
Issues:

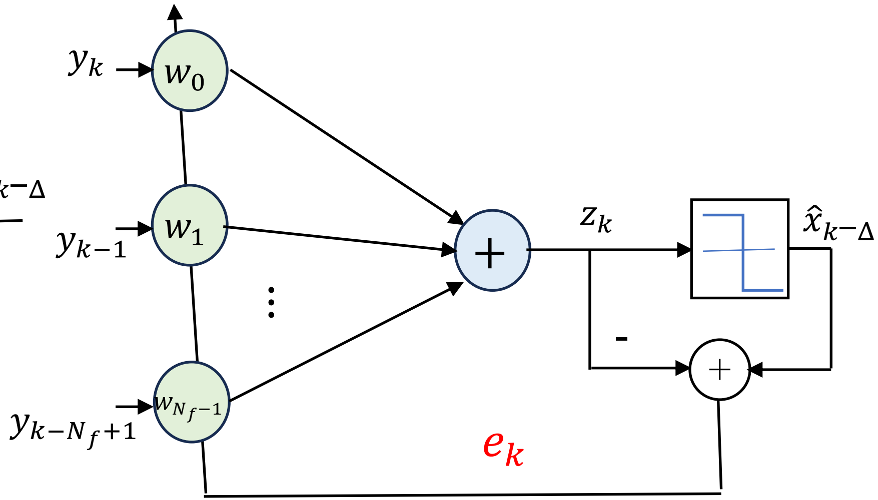
1. **Slight rate loss**
2. **Delay of packet if outer code used.**

Adaptive Equalizers & Channel ID

Section 3.14

Continuous Adaptation





- The error sequence is generated via:
  - Known training data, in which case $\hat{x}_{k-\Delta} = x_{k-\Delta}$, or
  - Assuming decisions are correct ("decision-directed").
- The **Least Mean Square (LMS)** algorithm (Widrow-Hoff):

$$\boldsymbol{w}_{k+1} = \boldsymbol{w}_k - \frac{\mu}{2} \cdot \nabla_{\boldsymbol{w}_k} \mathbb{E}\left[|e_k|^2\right]$$

$$\nabla_{\boldsymbol{w}_k} |e_k|^2 = -2 \cdot e_k^* \cdot \boldsymbol{y}_k$$

$$\begin{aligned} e_k &= \hat{x}_{k-\Delta} - \boldsymbol{w}_k^* \cdot \boldsymbol{y}_k \\ \boldsymbol{w}_{k+1} &= \boldsymbol{w}_k + \mu \cdot e_k^* \cdot \boldsymbol{y}_k \end{aligned}$$

Orth principle – e, y orthogonal stops updating

Look familiar?
One neuron with ReLU.

Indeed ReLU's came from adaptive equalizers (early 60's).

DFE Version

- Use the same embedding concept
  - $\tilde{\boldsymbol{w}} \to [\boldsymbol{w} \quad \boldsymbol{b}]$
- Same update equation on $\tilde{\boldsymbol{w}}$.
- The **step size** , μ ?

$$\boxed{0 \leq \mu \leq \frac{1}{N_{ff} \cdot \mathcal{E}_{\boldsymbol{y}} + N_{fb} \cdot \mathcal{E}_{\boldsymbol{x}}}}$$

- Leaky LMS?
  - $\beta \approx .99$

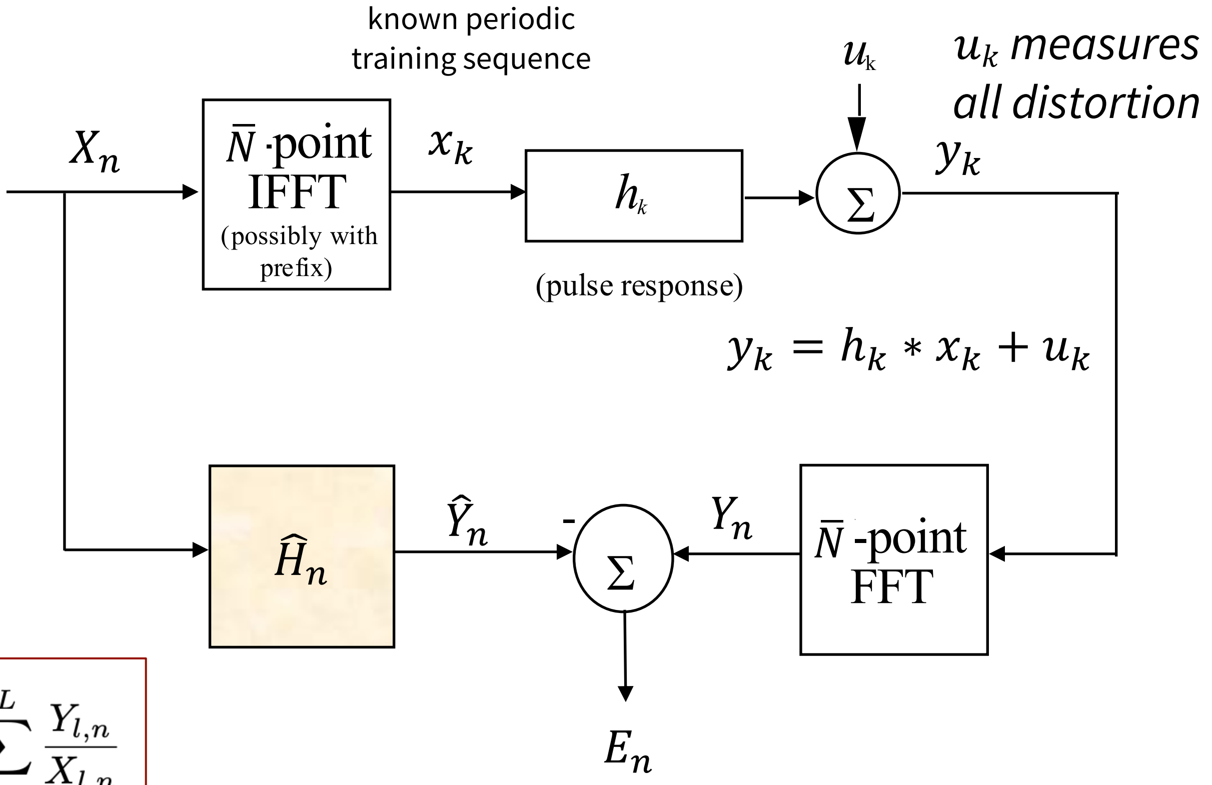
$$\boldsymbol{w}_{k+1} = \beta \cdot \boldsymbol{w}_k + \mu \cdot e_k^* \cdot \boldsymbol{y}_k$$

- This ensures stability.
- Difficult channels (those with severe ISI) cause the LMS to converge slowly.
- Convergence accelerates through weighting the gradient estimate by R_{yy}^{-1}.
  - But this means the receiver has to estimate R_{yy}^{-1}.
- It may be better to find $\boldsymbol{h}$ instead → **channel identification**.

Packet Adaptation

- Packet of $\bar{N}$ dimensions
  - In frequency domain here (no MIMO).
- Steepest descent and Hessian simplify
- Entire band is excited during training
  - Usually
- Just divide Output of rcvr FFT by known input and average
- The (usually periodic) training sequence repeats L times
  - Here assumed periodic

$$\hat{H}_n = \frac{1}{L} \sum_{l=1}^{L} \frac{Y_{l,n}}{X_{l,n}}$$



Norm Tap Error

- Parsevals $\|\boldsymbol{h}\|^2 = \|\boldsymbol{H}\|^2$
  - And all other frequency-domain/time-domain vectors.

$$\begin{aligned}
\text{NTE} &\triangleq E\left\{\|\boldsymbol{h} - \hat{\boldsymbol{h}}\|^2\right\} = E\left\{\|\boldsymbol{\delta}\|^2\right\} \\
&= E\left\{\|\boldsymbol{H} - \hat{\boldsymbol{H}}\|^2\right\} = E\left\{\|\boldsymbol{\Delta}\|^2\right\} \\
&= \sum_{n=0}^{\bar{N}-1} |H_n - \hat{H}_n|^2 = \sum_{k=0}^{\bar{N}-1} |h_k - \hat{h}_k|^2
\end{aligned}$$

$$\hat{H}_n = H_n + \sum_{l=1}^{L} \frac{U_{l,n}}{L \cdot X_{l,n}} \quad ,$$

$$\Delta_n = -\sum_{l=1}^{L} \frac{U_{l,n}}{L \cdot X_{l,n}} \quad .$$

$$\begin{aligned}
E_n &= Y_n - \hat{H}_n \cdot X_n = \Delta_n \cdot X_n + U_n \\
&= U_n + \frac{1}{L} \cdot \sum_{l=1}^{L} U_{l,n} \cdot e^{\jmath(\theta_n - \theta_{l,n})} \quad .
\end{aligned}$$

$$\boxed{SNR_{\hat{H},n} = \frac{R_{xx,n} \cdot |H_n|^2}{R_{ee,n} - R_{uu,n}} = \frac{SNR_n}{1/L} = L \cdot SNR\ (all\ n)}$$

Good training sequence?

$$x_k = e^{\jmath \frac{2\pi}{N} k^2}$$

- Same in all dimensions
  - The L =40 leaves 0.1 dB gain-estimation error

Noise Estimation

- Average the errors in frequency domain

$$\hat{\sigma_n^2} = \frac{1}{L} \cdot \sum_{l=1}^{L} |E_{l,n}|^2$$

$$\boxed{\mathrm{var}\left(\hat{\sigma_n^2}\right) = \frac{1}{L^2}\left(3 \cdot L \cdot \sigma_n^4 - L \cdot (\sigma_n^2)^2\right) = \frac{2}{L}\sigma_n^4}$$

- Noise miss only reduces with sqrt(L).

$$\sqrt{2/L} \cdot \sigma_n^2$$

STANFORD

End Lecture 16