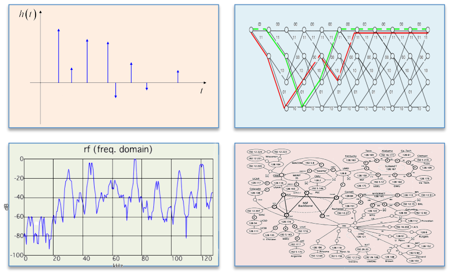


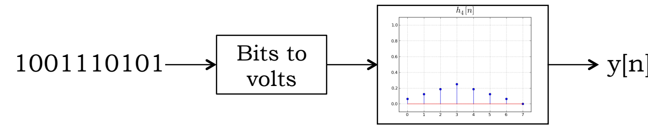
INTRODUCTION TO EECS II

DIGITAL COMMUNICATION SYSTEMS

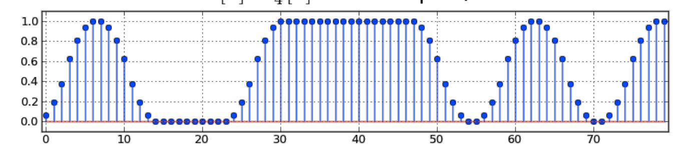
6.02 Spring 2011
Lecture #5

- Intersymbol interference
- Deconvolution
- Stability & noise, approximate deconvolvers

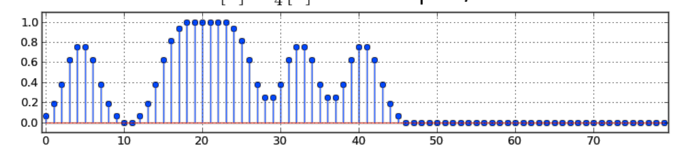
Transmission Over a Channel



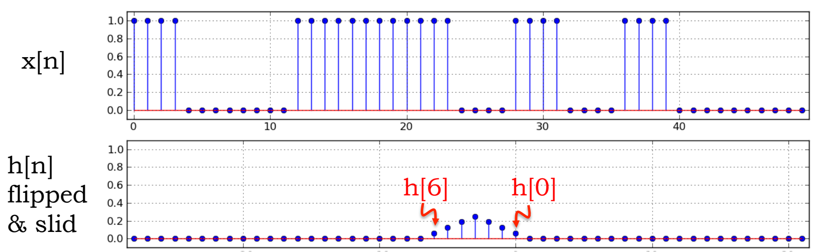
$x[n] * h_4[n]$ at 8 samples/bit



$x[n] * h_4[n]$ at 4 samples/bit



Convolution sum: "flip and slide"

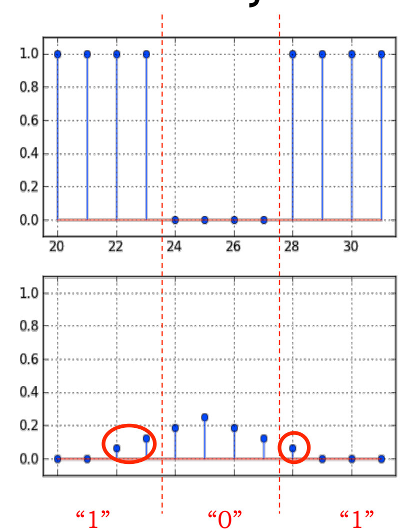


y[28] = x[28]h[0] + x[27]h[1] + … + x[22]h[6]

Visual representation of convolution sum: do a horizontal flip of the of graph of h[n], then slide along under x[n].

To compute y[m], slide flipped h[n] until h[0] is under x[m], then compute sum of element-by-element product of the two sequences.

Intersymbol Interference (ISI)



Issue:
If we send a small number of samples/bit, the active portion of h[n] may cover more than one bit cell when doing convolution sum.

Result:
y[n] values for a particular bit cell include contributions from neighboring cells.

Example: y[28] is the lowest voltage received for the "0" bit, but includes contributions from the neighboring "1" bits.

Given h[n], how bad is ISI?

<u>Recipe:</u>

1. Compute B, the number bits "covered" by h[n]. Let N = samples/bit

$$B = \left\lfloor \frac{\text{length of active portion of h[n]}}{\text{N}} \right\rfloor + 2$$

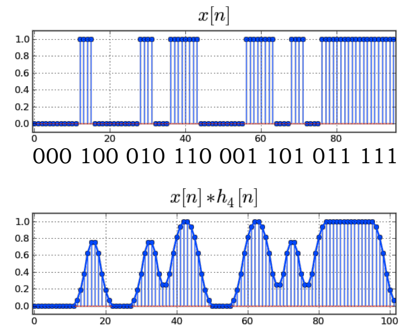
2. Generate a test pattern that contains all possible combinations of B bits – want all possible combinations of neighboring cells. If B is big, randomly choose a large number of combinations.

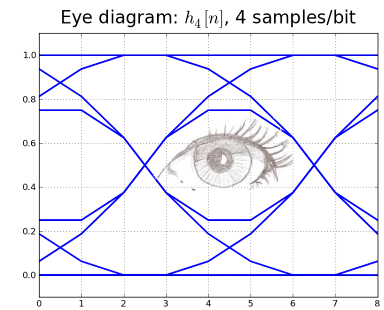
3. Transmit the test pattern over the channel (2^N*B samples)

4. Instead of one long plot of y[n], plot the response as an *eye diagram:*
   a. break the plot up into short segments each containing 2N+1 samples, starting at sample 0, N, 2N, 3N, …
   b. plot all the short segments on top of each other

Eye Diagram Example

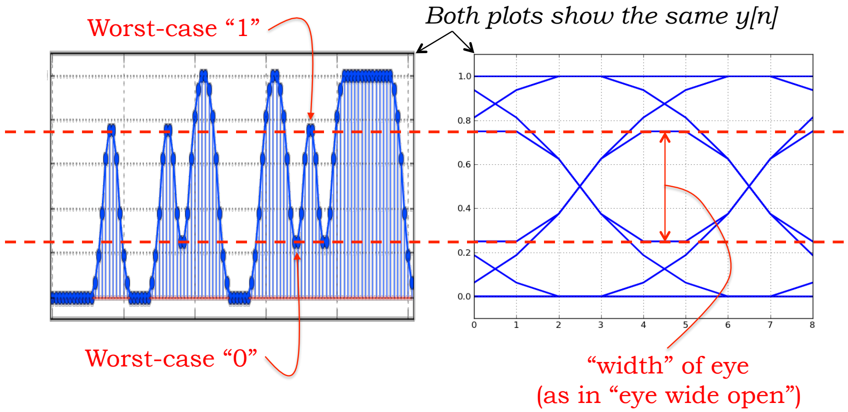
Using h_4[n] and samples_per_bit=4: B = 3





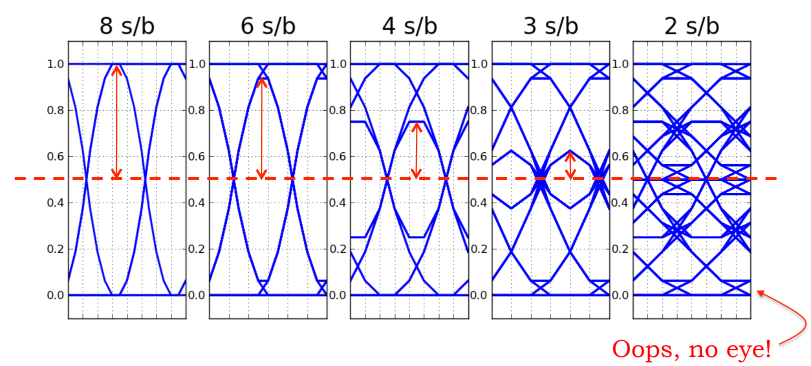
Eye diagrams make it easy to find the worst-case signaling conditions at the receiving end.

"Width" of Eye



To maximize noise margins:
Pick the best sample point → widest point in the eye
Pick the best digitization threshold → half-way across width

Choosing Samples/Bit

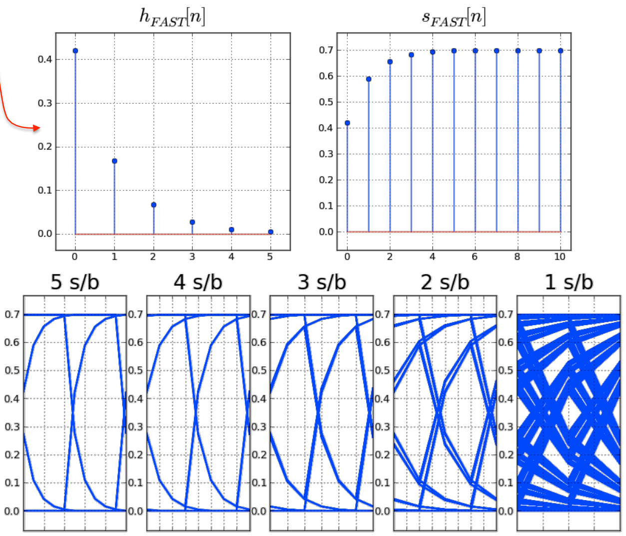


Given h[n], you can use the eye diagram to pick the number of samples transmitted for each bit (N):

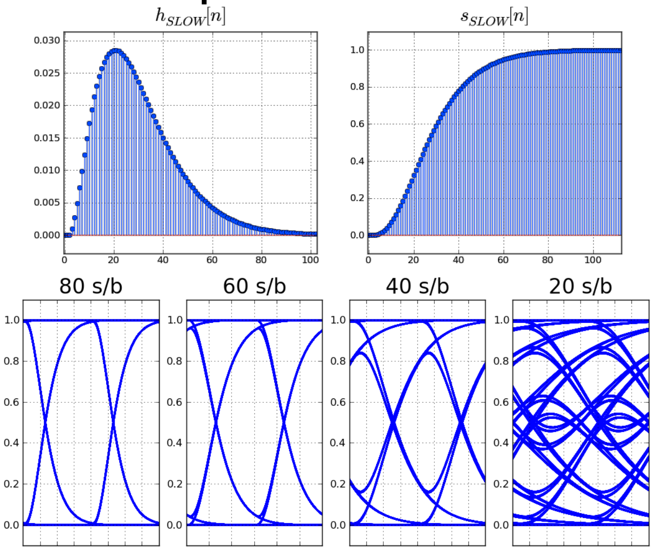
Reduce N until you reach the noise margin you feel is the minimum acceptable value.

Example: "fast" channel

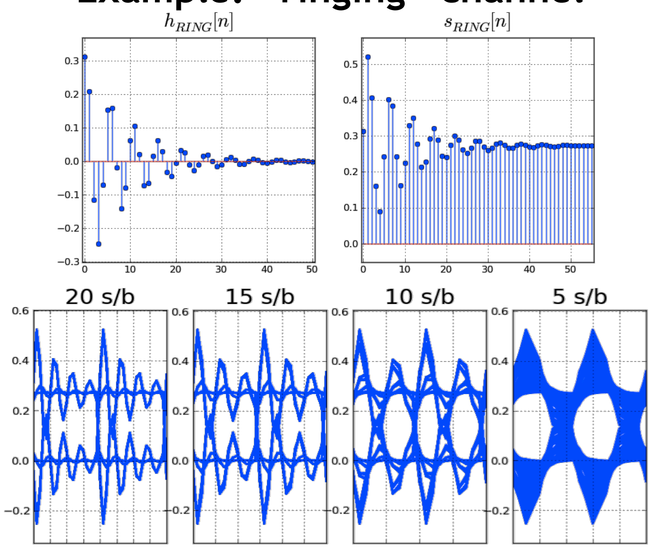
h[n] for actual for actual channels extend to +∞, although values get vanishing small. For practicality, we truncate when terms fall below a certain value.



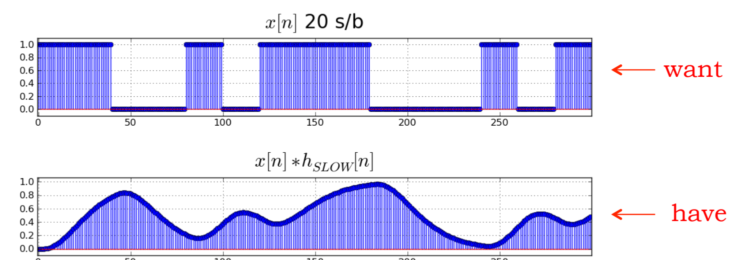
Example: "slow channel"



Example: "ringing" channel



Can We Recover From ISI?



After all, in a perfect world (no noise), no information has been lost, only spread out over many samples.

Given y[n] and h[n], can we develop an estimate w[n] for the actual input waveform x[n]? We could, of course, easily receive x[n]!

Difference Equation for w[n]

If w[n] was a perfect estimate of x[n], it would satisfy:

$$y[n] = w[n]h[0] + w[n-1]h[1] + w[n-2]h[2] + \ldots + w[n-K]h[K]$$

Simplifying assumption: h[K] is last non-zero element

Let's solve this for w[n]:

$$w[n] = \frac{1}{h[0]}\Big(y[n] - \big(w[n-1]h[1] + w[n-2]h[2] + \ldots + w[n-K]h[K]\big)\Big)$$

$$\boxed{\begin{array}{ll} h[i]=0 & i<0 \text{ or } i>K \\ w[j]=0 & j<0 \end{array}}$$

Given y[n] and h[n], we can incrementally compute sequence w[n] using a straightforward "plug and chug" approach:

$$w[0] = \frac{1}{h[0]}\big(y[0]\big)$$

$$w[1] = \frac{1}{h[0]}\big(y[1] - w[0]h[1]\big)$$

$$w[2] = \frac{1}{h[0]}\big(y[2] - w[1]h[1] - w[0]h[2]\big)$$

What if h[0]=0?

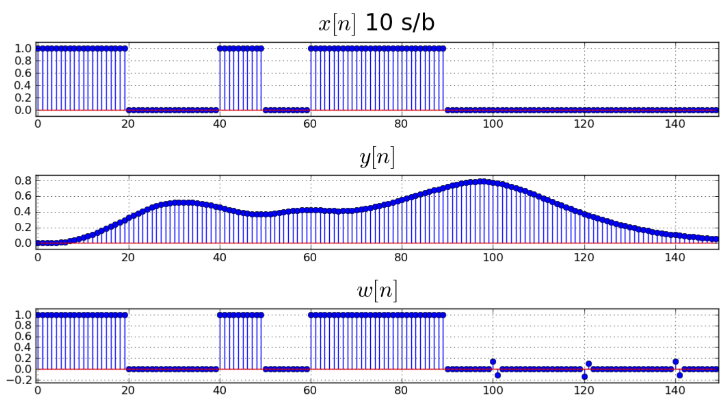
$$w[n] = \frac{1}{h[0]}\Big(y[n] - \big(w[n-1]h[1] + w[n-2]h[2] + \ldots + w[n-K]h[K]\big)\Big)$$

Oops! Division by 0 isn't a good idea…

Zeros at the beginning h[n] represent a channel with a delay: m zeros would mean a m-sample delay. We can eliminate the delay without affecting our estimate for x[n]. So

1. Count the number of zeros at the front of h[n] = m
2. Eliminate the first m elements of h[n], and eliminate the first m elements of y[n]
3. Now use the equation above on the shortened h[n] and y[n]

Deconvolution Example



???
(hint: see slide #10)

Sensitivity to Noise

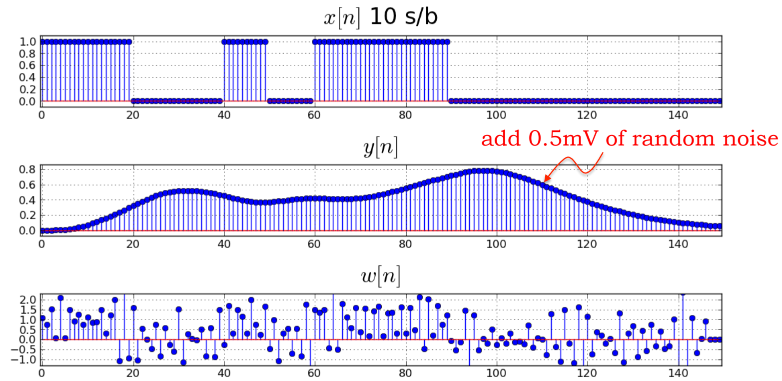
Let's consider what happens if some small amount of noise (ε) is added to the first sample of the response (y[0]):

Estimate	Error
$w[0] = \frac{1}{h[0]}\big(y[0] + \varepsilon\big)$	$\frac{\varepsilon}{h[0]}$
$w[1] = \frac{1}{h[0]}\big(y[1] - w[0]h[1]\big)$	$-\frac{\varepsilon}{h[0]}\frac{h[1]}{h[0]}$
$w[2] = \frac{1}{h[0]}\big(y[2] - w[1]h[1] - w[0]h[2]\big)$	$\frac{\varepsilon}{h[0]}\left(\frac{h[1]}{h[0]}\right)^2 - \frac{\varepsilon}{h[0]}\frac{h[2]}{h[0]}$

Question: is the error growing as we compute more w's?

Answer: depends on h[0] and the ratios h[m]/h[0]. Small values of h[0] and (h[m]/h[0]) > 1 are troublesome…

Noisy Deconvolution Example



Urk!

Stability Criterion

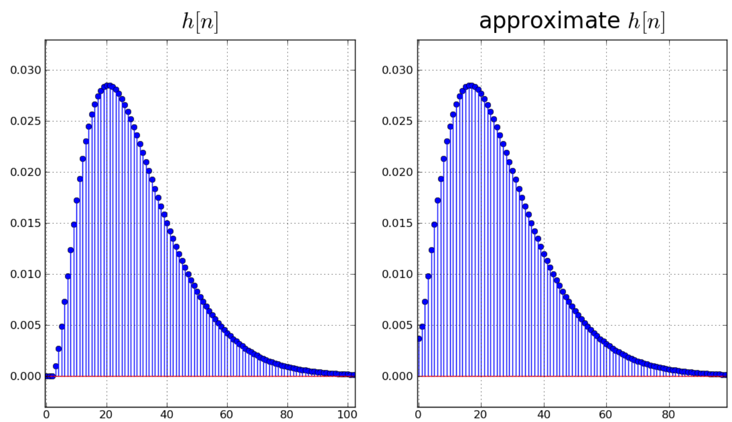
The notes have a derivation of the following sufficient (very conservative) condition that will ensure the stability of the deconvolver operating on a noisy y[n]:

$$\sum_{m=1}^{K}\left|\frac{h[m]}{h[0]}\right| < 1 \quad \text{or, perhaps more usefully} \quad \sum_{m=1}^{K}\left|h[m]\right| < \left|h[0]\right|$$

What if my h[n] doesn't meet this criterion?

Make a new "approximate" h[n] that does! Combine samples at the beginning of h[n] to make a bigger h[0].

Example Approximate $h_{SLOW}[n]$



Approximation: combine first 5 samples of $h_{SLOW}[n]$

(Less) Noisy Deconvolution Example