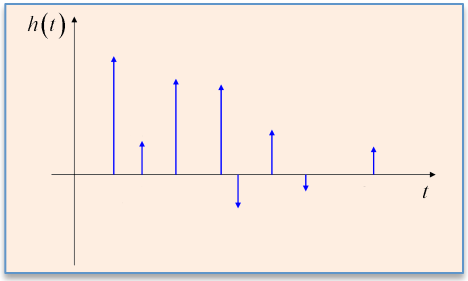
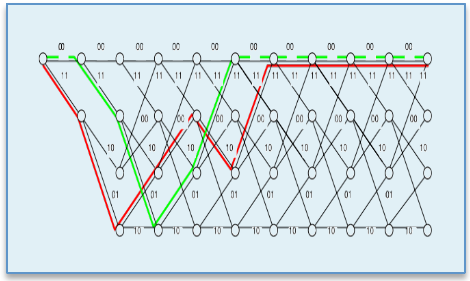
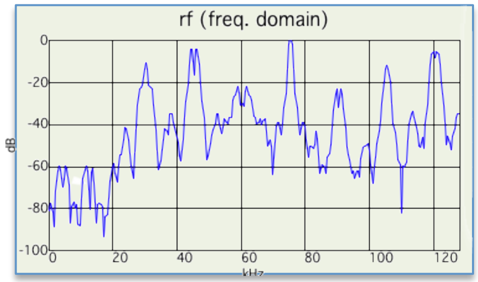
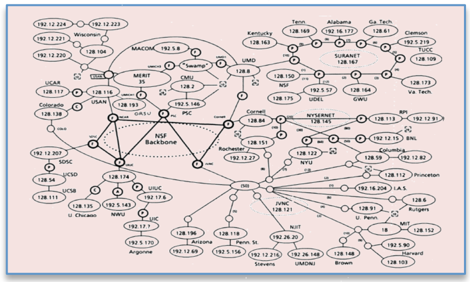








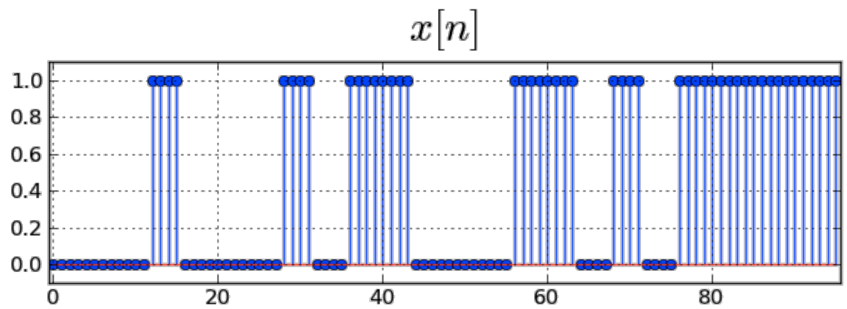
INTRODUCTION TO EECS II

DIGITAL COMMUNICATION SYSTEMS

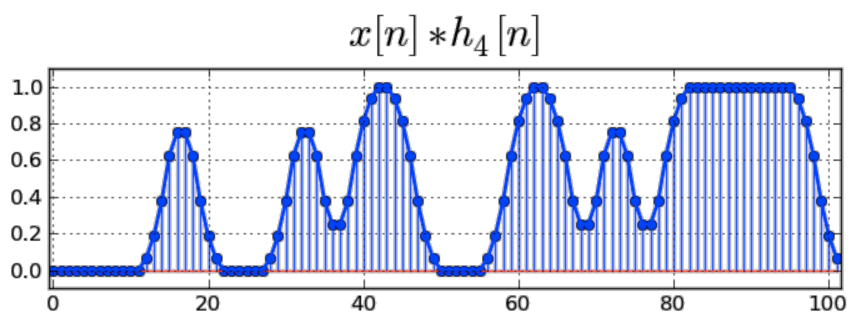
6.02 Fall 2012
Lecture #11

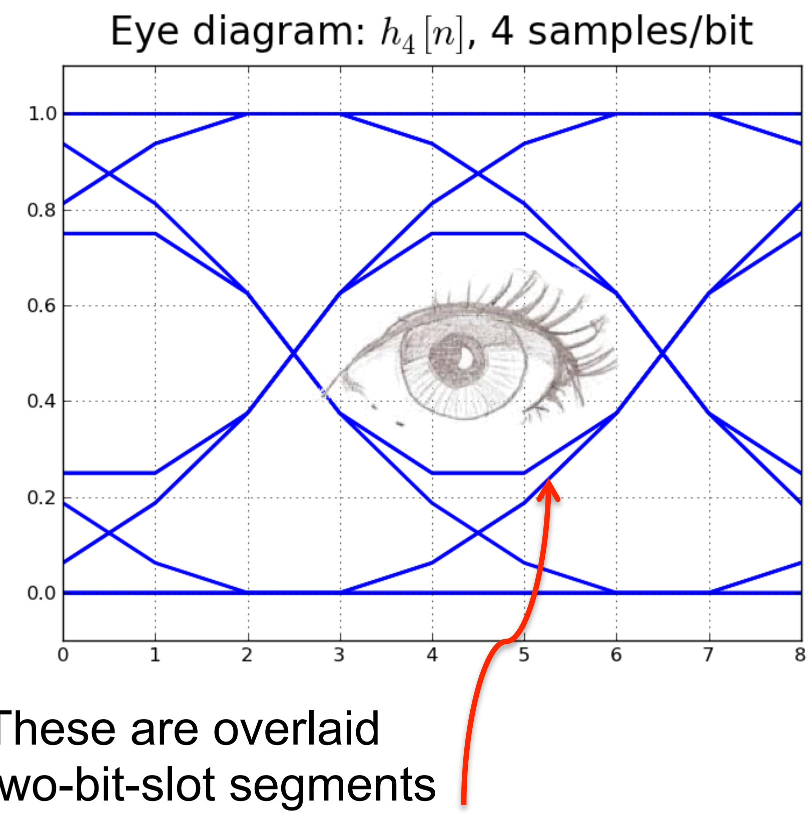
- Eye diagrams
- Alternative ways to look at convolution

Eye Diagrams



000 100 010 110 001 101 011 111

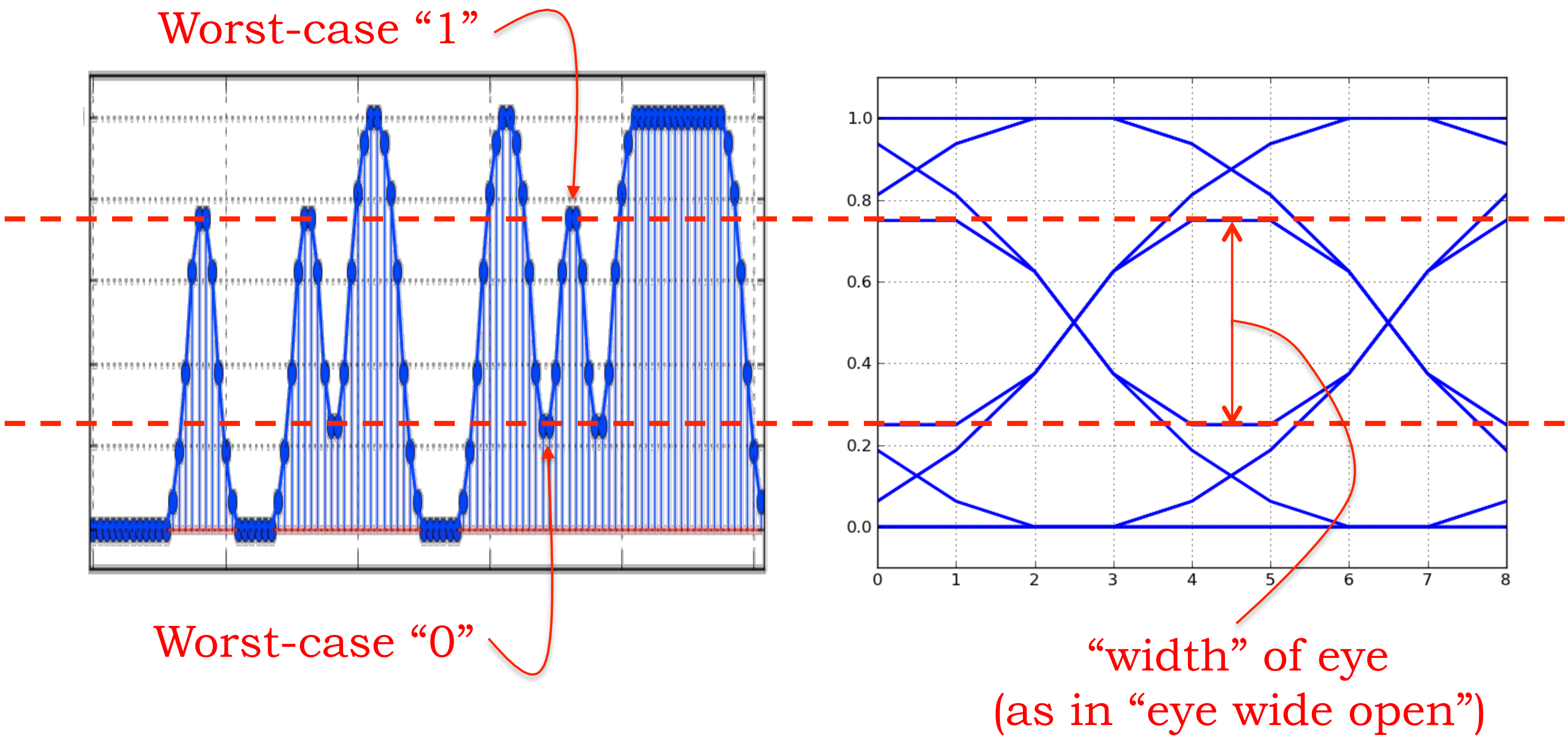




Eye diagrams make it easy to find the worst-case signaling conditions at the receiving end.

These are overlaid two-bit-slot segments of step responses, plotted without the 'stems' of the stem plot on the left

“Width” of Eye

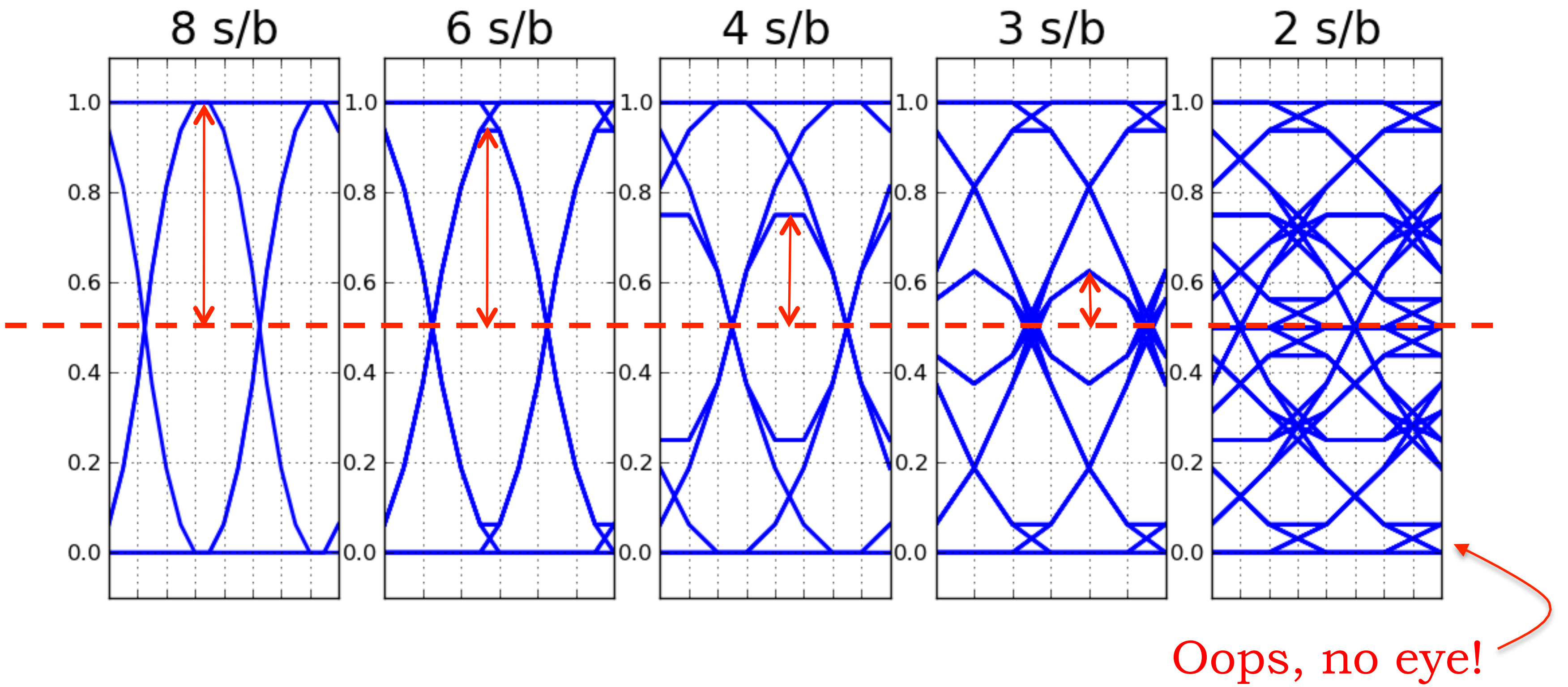


To maximize noise margins:

Pick the best sample point → widest point in the eye

Pick the best digitization threshold → half-way across width

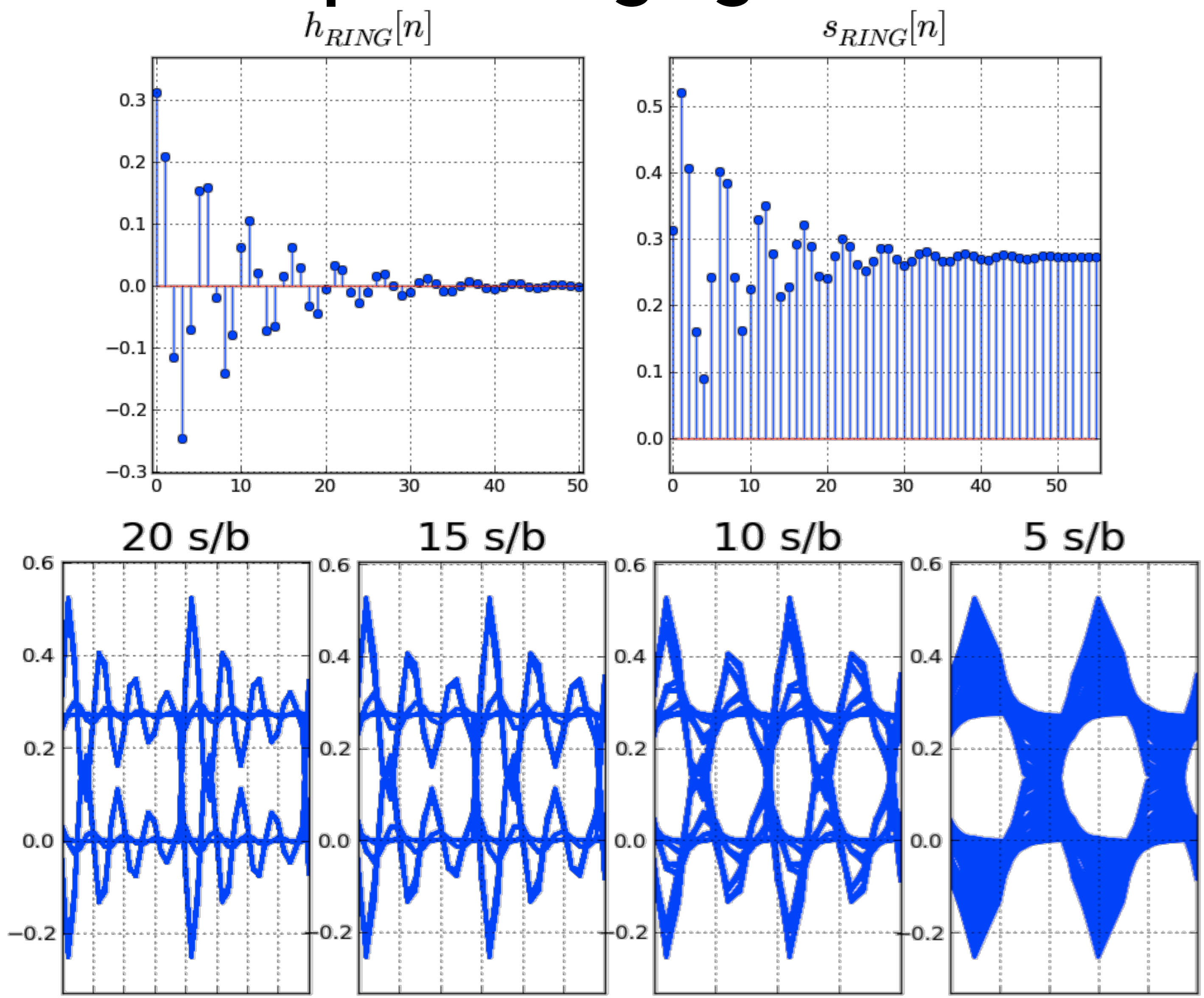
Choosing Samples/Bit



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: “ringing” channel



Constructing the Eye Diagram (no need to wade through all this unless you really want to!)

1. Generate an input bit sequence pattern that contains all possible combinations of B bits (e.g., B=3 or 4), so a sequence of $2^B B$ bits. (Otherwise, a random sequence of comparable length is fine.)

2. Transmit the corresponding x[n] over the channel ($2^B BN$ samples, if there are N samples/bit)

3. 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 KN samples, starting at sample 0, KN, 2KN, 3KN, … (e.g., K=2 or 3)
   b. plot all the short segments on top of each other

Back To Convolution

From last lecture: If system S is both linear and time-invariant (LTI), then we can use the unit sample response h[n] to predict the response to *any* input waveform x[n]:

Sum of shifted, scaled unit sample functions

Sum of shifted, scaled unit sample *responses*, with the same scale factors

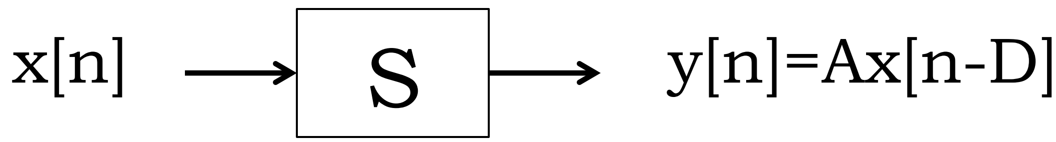
$$x[n] = \sum_{k=-\infty}^{\infty} x[k]\delta[n-k] \longrightarrow \boxed{S} \longrightarrow y[n] = \sum_{k=-\infty}^{\infty} x[k]h[n-k]$$

CONVOLUTION SUM

Indeed, the unit sample response h[n] completely characterizes the LTI system S, so you often see

$$x[n] \longrightarrow \boxed{h[.]} \longrightarrow y[n]$$

Unit Sample Response of a Scale-&-Delay System



If S is a system that scales the input by A and delays it by D time steps (negative ‘delay’ D = advance), is the system

time-invariant? Yes!

linear? Yes!

Unit sample response is $h[n]=A\delta[n-D]$

General unit sample response

$$h[n]=\ldots + h[-1]\delta[n+1] + h[0]\delta[n] + h[1]\delta[n-1]+\ldots$$

for an LTI system can be thought of as resulting from many scale-&-delays in parallel

A Complementary View of Convolution

So instead of the picture:

$$x[n] = \sum_{k=-\infty}^{\infty} x[k]\delta[n-k] \longrightarrow \boxed{h[.]} \longrightarrow y[n] = \sum_{k=-\infty}^{\infty} x[k]h[n-k]$$

we can consider the picture:

$$x[n] \longrightarrow \boxed{h[.] = \ldots + h[-1]\delta[n+1] + h[0]\delta[n] + h[1]\delta[n-1] + \ldots} \longrightarrow y[n]$$

from which we get $$y[n] = \sum_{m=-\infty}^{\infty} h[m]x[n-m]$$

(To those who have an eye for these things, my apologies for the varied math font --- too hard to keep uniform!)

(side by side)

$$y[n] =$$

$$(x * h)[n] = \sum_{k=-\infty}^{\infty} x[k]h[n-k] \quad = \quad \sum_{m=-\infty}^{\infty} h[m]x[n-m] = (h * x)[n\ ]$$

Input term x[0] at time 0 launches scaled unit sample response x[0]h[n] at output

Unit sample response term h[0] at time 0 contributes scaled input h[0]x[n] to output

Input term x[k] at time k launches scaled shifted unit sample response x[k]h[n-k] at output

Unit sample response term h[m] at time m contributes scaled shifted input h[m]x[n-m] to output

To Convolve (but not to "Convolute"!)

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

A simple graphical implementation:

Plot x[.] and h[.] as a function of the dummy index (k or m above)

Flip (i.e., reverse) one signal in time, **slide** it right by n (slide left if n is –ve), take the **dot.product** with the other.

This yields the value of the convolution at the single time n.

'flip one & slide by n …. dot.product with the other'

Example

- From the unit sample response h[n] to the unit step response

$$s[n] = (h * u)[n]$$

- Flip u[k] to get u[-k]
- Slide u[-k] n steps to right (i.e., delay u[-k]) to get u[n-k]), place over h[k]
- Dot product of h[k] and u[n-k] wrt k:

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

Channels as LTI Systems

Many transmission channels can be effectively modeled as LTI systems. When modeling transmissions, there are few simplifications we can make:

- We'll call the time transmissions start t=0; the signal before the start is 0. So x[m] = 0 for m < 0.
- Real-word channels are ***causal:*** the output at any time depends on values of the input at only the present and past times. So h[m] = 0 for m < 0.

These two observations allow us to rework the convolution sum when it's used to describe transmission channels:

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

start at t=0 causal j=n-k

Properties of Convolution

$$(x*h)[n] \equiv \sum_{k=-\infty}^{\infty} x[k]h[n-k] = \sum_{m=-\infty}^{\infty} h[m]x[n-m]$$

The second equality above establishes that convolution is commutative:

$$x*h = h*x$$

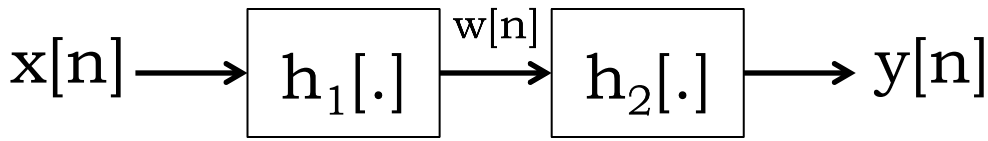
Convolution is associative:

$$x*(h_1*h_2) = \left(x*h_1\right)*h_2$$

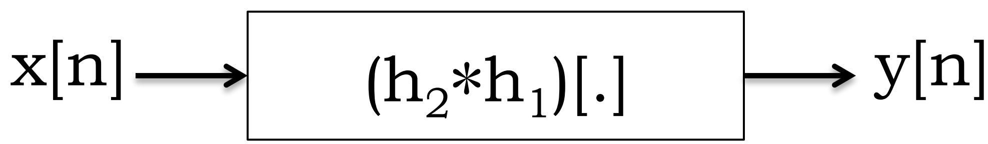
Convolution is distributive:

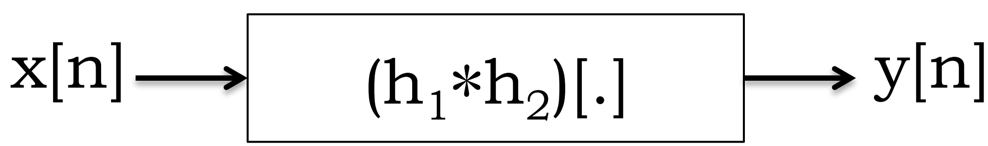
$$x*\left(h_1+h_2\right) = (x*h_1)+(x*h_2)$$

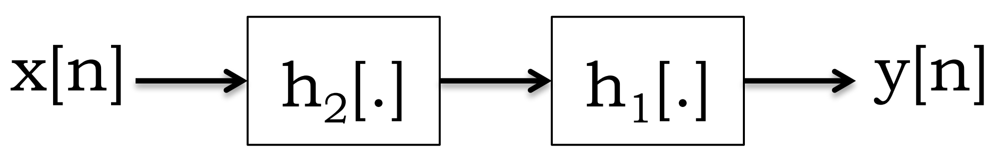
Series Interconnection of LTI Systems



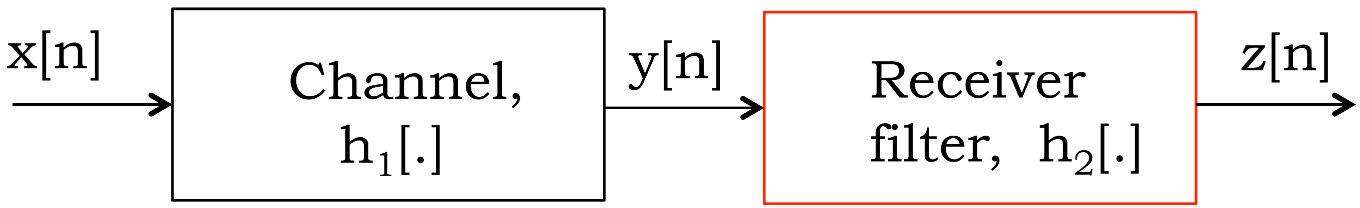
$$y = h_2 * w = h_2 * \left(h_1 * x\right) = \left(h_2 * h_1\right) * x$$







“Deconvolving” Output of Echo Channel



Suppose channel is LTI with

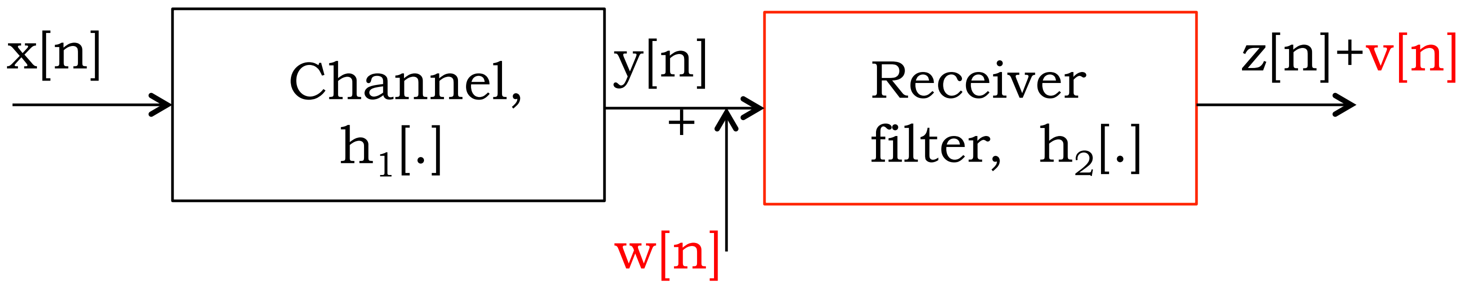
$$h_1[n]=\delta[n]+0.8\delta[n-1]$$

Find $h_2[n]$ such that $z[n]=x[n]$

$$\Rightarrow \quad (h_2 * h_1)[n]=\delta[n]$$

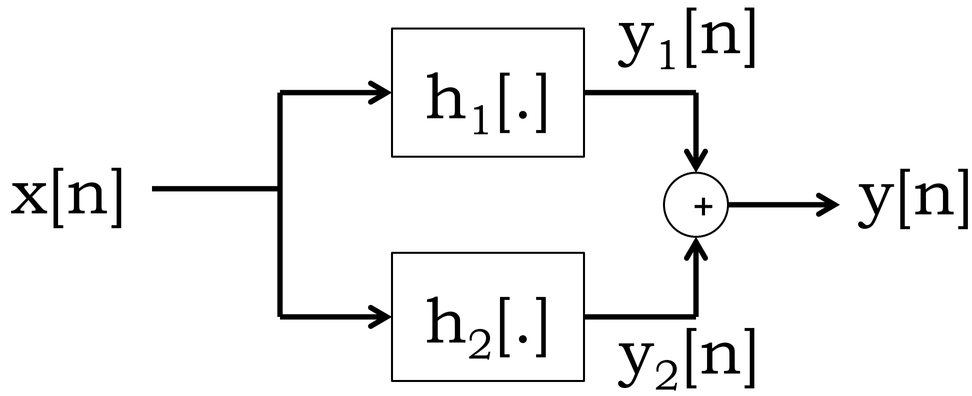
Good exercise in applying Flip/Slide/Dot.Product

"Deconvolving" Output of Channel with Echo



Even if channel was well modeled as LTI and $h_1[n]$ was known, noise on the channel can greatly degrade the result, so this is usually not practical.

Parallel Interconnection of LTI Systems



$$y = y_1 + y_2 = (h_1 * x) + (h_2 * x) = \left(h_1 + h_2\right) * x$$