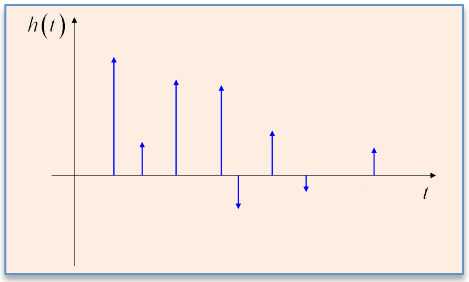
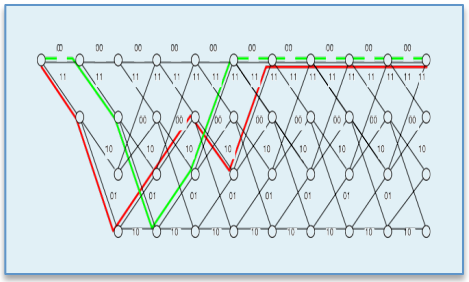
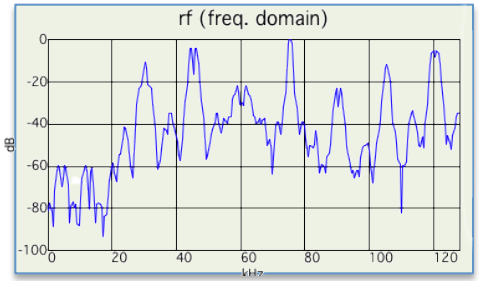
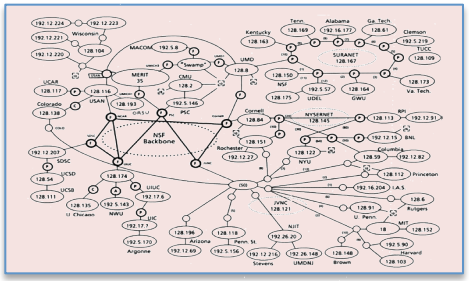








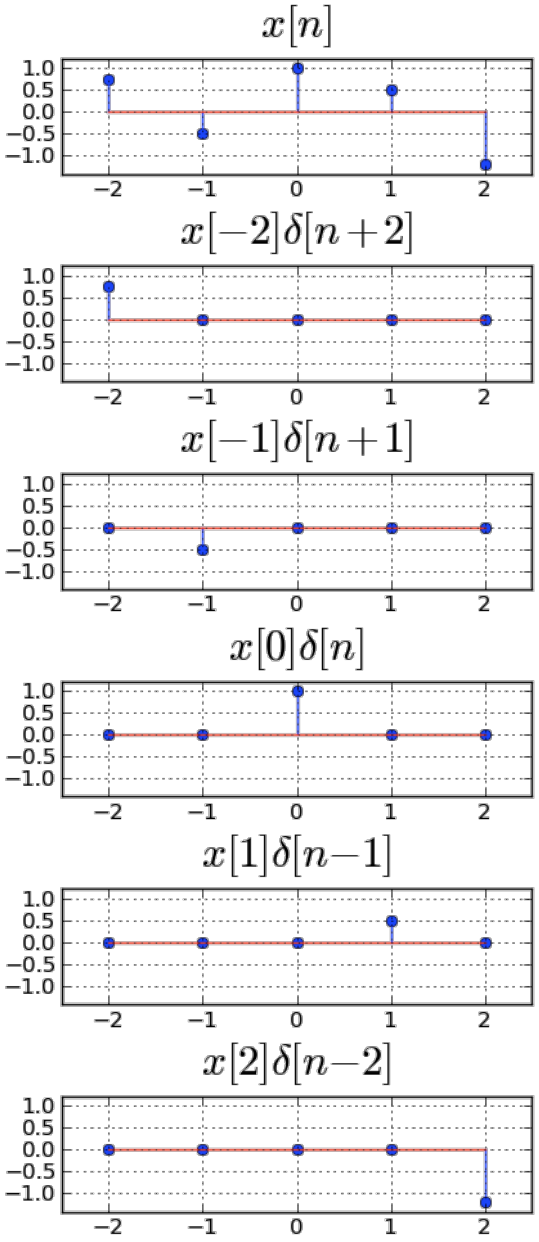
INTRODUCTION TO EECS II

DIGITAL COMMUNICATION SYSTEMS

6.02 Fall 2013
Lecture #11

- Convolution

Unit Sample Decomposition



A discrete-time signal can be decomposed into a sum of time-shifted, scaled unit samples.

Example: in the figure, x[n] is the sum of x[-2]δ[n+2] + x[-1]δ[n+1] + … + x[2]δ[n-2].

In general:

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

For any particular index, only one term of this sum is non-zero

Modeling LTI Systems

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

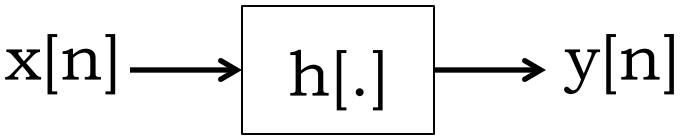
Sum of shifted, scaled unit samples

Sum of shifted, scaled responses

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

CONVOLUTION SUM

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



Convolution

Evaluating the convolution sum

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

for all n defines the output signal y in terms of the input x and unit-sample response h. Some constraints are needed to ensure this infinite sum is well behaved, i.e., doesn't "blow up" --- we'll discuss this later.

We use $*$ to denote convolution, and write y=x$*$h. We can then write the value of y at time n, which is given by the above sum, as

$$y[n] = (x * h)[n] \qquad \text{or} \qquad y[n] = x * h[n]$$

Instead you'll find people writing $y[n] = x[n] * h[n]$, where the poor index n is doing double or triple duty. This is lousy notation, but a super-majority of engineering professors (including at MIT) will inflict it on their students.

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:

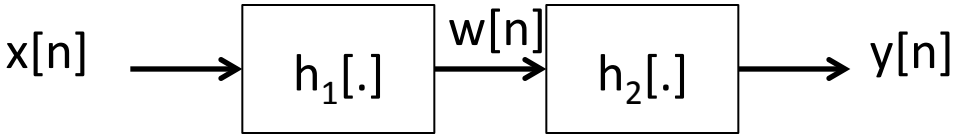
$$x * (h_1 * h_2) = \left(x * h_1\right) * h_2$$

Convolution is distributive:

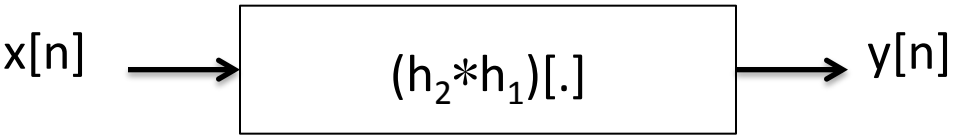
$$x * \left(h_1 + h_2\right) = (x * h_1) + (x * h_2)$$

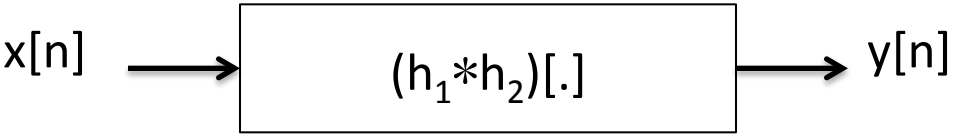
(Some conditions needed, typically guaranteeing that the individual convolutions are well-behaved.)

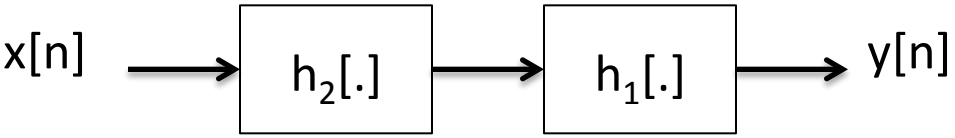
Series Interconnection of LTI Systems



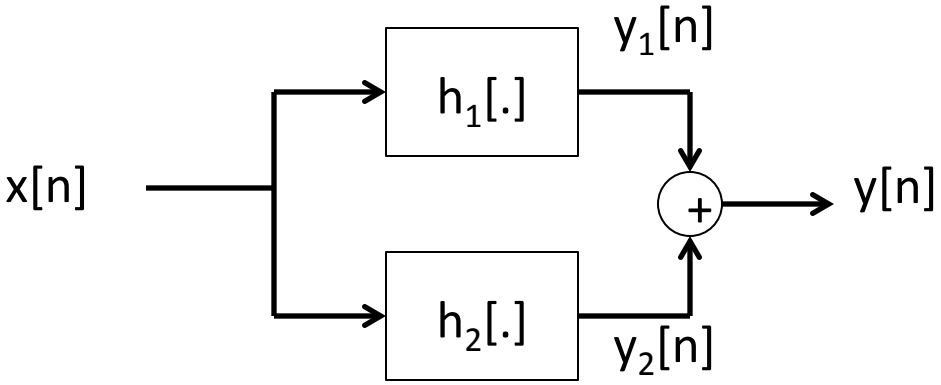
$$y = h_2 * w = h_2 * \left(h_1 * x\right) = \left(h_2 * h_1\right) * x$$



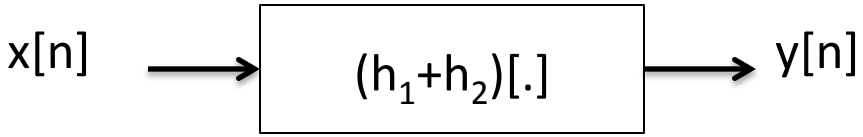




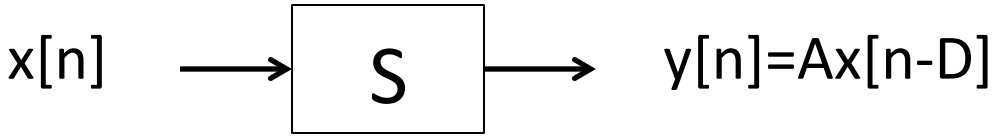
Parallel Interconnection of LTI Systems



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



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

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

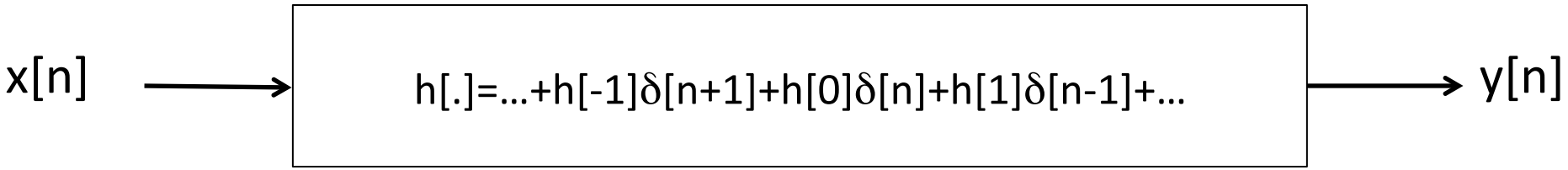
for an LTI system can be thought of as resulting from many scale-&-delays in parallel

A Complementary View of Convolution

So instead of the picture:

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

we can consider the picture:



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"!)

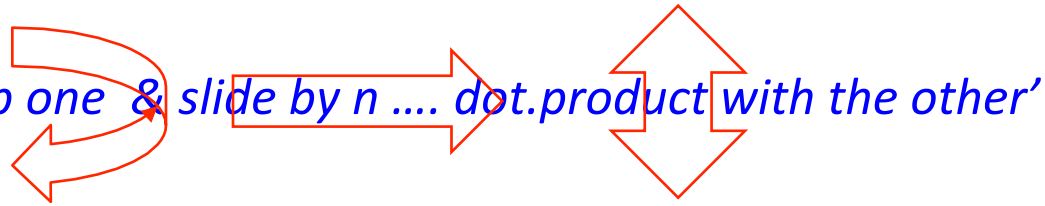
$$\sum_{k=-\infty}^{\infty} x[k]h[n-k] = \sum_{m=-\infty}^{\infty} h[m]x[n-m]$$

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.



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:

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

Bounded-Input Bounded-Output (BIBO) Stability

What ensures that the infinite sum

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

is well-behaved?

One important case: If the unit sample response is *absolutely summable,* i.e.,

$$\sum_{m=-\infty}^{\infty} |h[m]| < \infty$$

and the input is *bounded,* i.e., $|x[k]| \leq M < \infty$

Under these conditions, the convolution sum is well-behaved, and the *output* is guaranteed to be *bounded*.

The absolute summability of h[n] is necessary and sufficient for this bounded-input bounded-output (BIBO) stability.

Channels as LTI Systems

Many transmission channels can be effectively modeled as LTI systems. When modeling transmissions, there area 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:

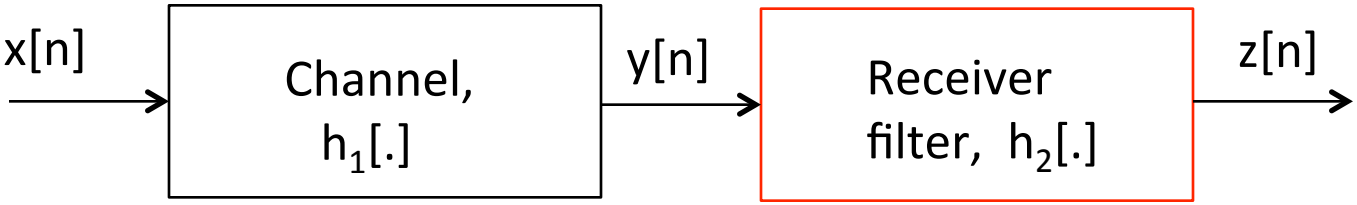
$$y[n] = \sum_{k=-\infty}^{\infty} x[k]h[n-k] = \sum_{k=0}^{\infty} x[k]h[n-k] = \sum_{k=0}^{n} x[k]h[n-k] = \sum_{j=0}^{n} x[n-j]h[j]$$

start at t=0

causal

j=n-k

"Deconvolving" Output of Echo Channel



Suppose channel is LTI with

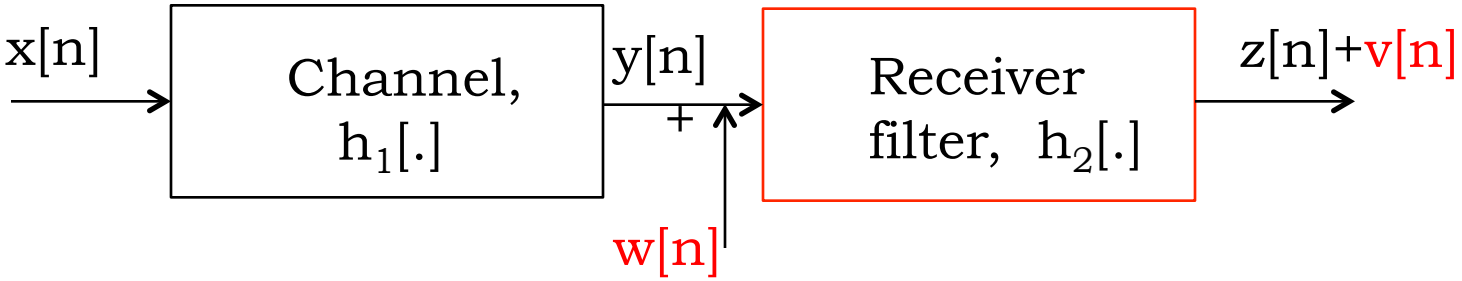
$$h_1[n]=\delta[n]+0.8\delta[n-1]$$

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

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

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.