MIT 6.02 **DRAFT** *Lecture Notes*
Spring 2010 (Last update: September 15, 2010)
Comments, questions or bug reports?
Please contact 6.02-staff@mit.edu

LECTURE 1
Undoing ISI with Deconvolution

This lecture shows how we can use the properties of causality and LTI system to undo channel-induced inter-symbol interference (ISI) in the sequence of received samples, and reconstruct the sequence of samples (and therefore bits) sent by the transmitter. In this lecture we will assume that the channel non-idealities only cause ISI and that noise. is absent. Of course, assuming no noise is an unrealizable idealization, and we will consider the impact of noise in subsequent lectures. The technique we will develop is referred to as **deconvolution**, and the basic idea is used in a wide range of applications outside of communication systems including medical imaging, microscopy, control and acoustic equalization.

■ 1.1 The Problem

Recall that our goal is to develop a processing module that will take as input the received samples and produce a set of samples that more closely resemble the transmitted samples (albeit possibly delayed), thereby making it possible to more reliably determine the transmitted bits. The processing module we will develop is based on a technique referred to as deconvolution, and for a causal LTI channel with no noise, deconvolution perfectly reconstructs the transmitted samples. In theory, this would mean that if we use deconvolution, then we can always use just 1 sample per bit regardless of the channel. In practice, the performance of unmodified deconvolution degrades rapidly with noise and numerical roundoff errors, making it quite difficult to develop robust communication systems using just few samples per bit. But, we will ignore the issue of noise for today, and return to it next week.

To understand the problem, we will go back to the notion of the eye diagram. Figure 1-1 shows several eye diagrams for a given channel, where different numbers of samples per bit were used. The picture on the top left shows the eye diagram with 50 samples per bit, while the one on the bottom left is for 5 samples per bit. The first one has a noticeable eye, but it isn't particularly wide, while the second one has no eye at all (except maybe in Picasso's eyes!). The beautiful outcome of the method we'll study in this lecture are the pictures on the top and bottom right, which show wide eyes. We want our technique to

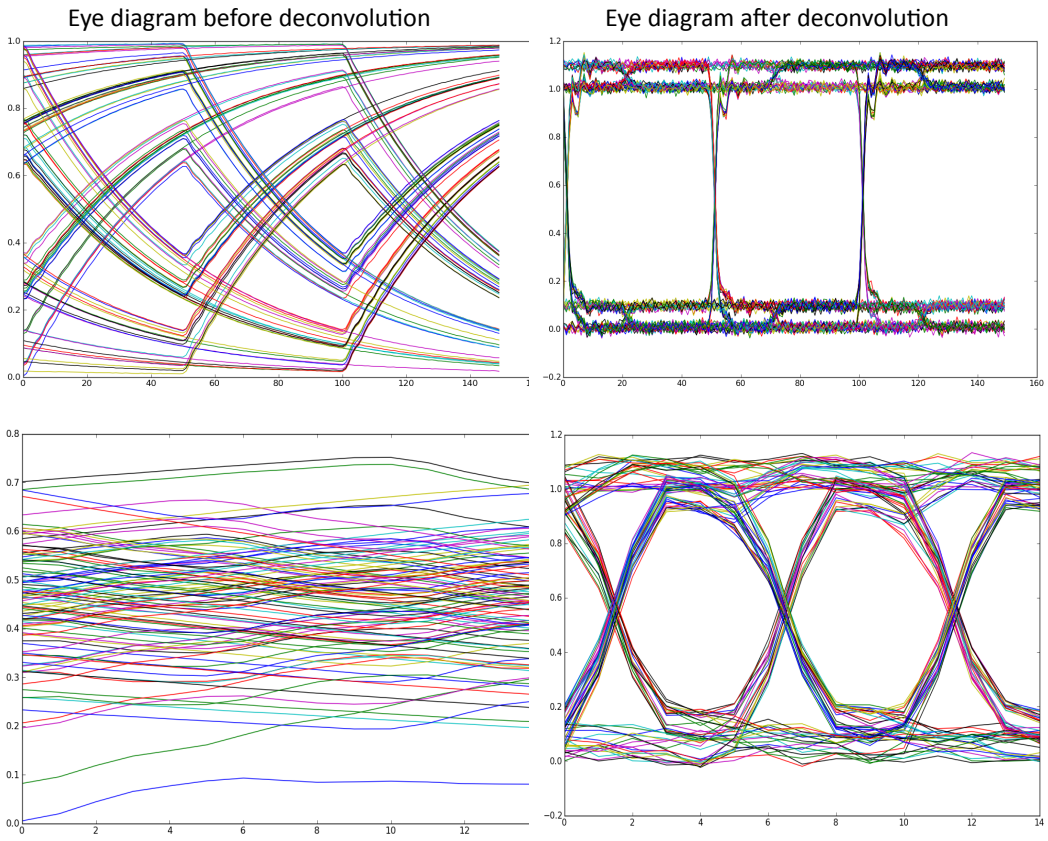

Figure 1-1: Eye diagrams before and after deconvolution for the 6.02 IR channel. The top pictures are for a larger number of samples per bit (50) compared to the bottom (5).

transform the "shut eyes" on the left into the "wide open" ones on the right, for that will enable us to overcome ISI and receive data with essentially no errors.

■ 1.2 Our Plan to Undo ISI

Our plan builds on the three properties of the communication link model we discussed in the last lecture: causality, linearity, and time-invariance. Our plan is as follows. First, we will characterize a causal LTI channel using the idea of a **unit sample response**, a concept you have already seen in 6.01. By applying the principle of superposition, we can determine the sequence of samples at the receiver end of the channel, Y, for *any* transmitted sequence X if we know the channel's unit sample response, H. The process of computing Y given X using H is referred to as **convolution**.

Then, given a sequence of output samples, Y, we will "reverse" the convolution to produce an estimate, W, of the input signal, X. This process is called **deconvolution** and is the central idea in undoing the effects of ISI.

The rest of this lecture will discuss the salient features of a causal LTI channel and explain why the unit sample response completely characterizes such a channel. Then, we

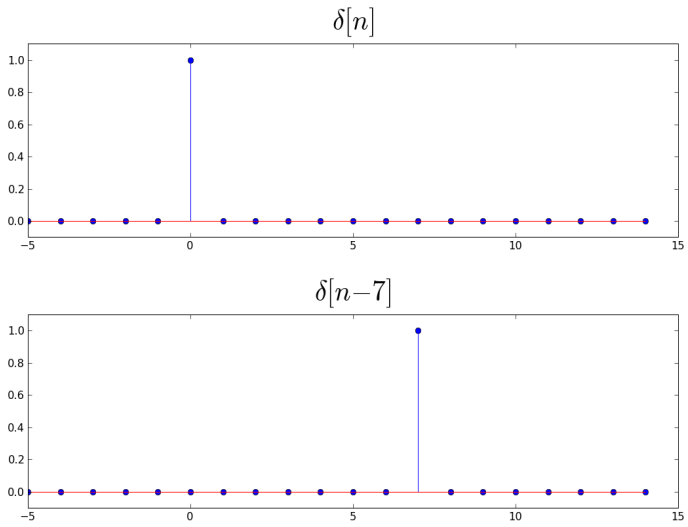

Figure 1-2: A unit sample. In the top picture the unit sample is unshifted, and is therefore nonzero only at 0, while the bottom picture is a unit sample shifted forward in time by 7 units (shifting forward in time means that we use the $-$ sign in the argument, since we want the "spike" to occur when $n - 7 = 0$.

will discuss how deconvolution works.

1.3 Understanding a Causal LTI Channel

A unit sample is a "spike" input, shown in Figure 1-2. Formally, a unit sample δ is defined as follows:

$$\begin{aligned} \delta[n] &= 1 \text{ if } n = 0 \\ \delta[n] &= 0 \text{ otherwise} \end{aligned} \tag{1.1}$$

The channel's response to a unit sample is referred to as the **unit sample response**, and is denoted H. H is a sequence of values, $h[0], h[1], \ldots, h[n], \ldots$, and represents the sequence of samples at the receiver end of the channel, given the transmitter generated a unit sample at the transmitter end of the channel. To both simplify the description, and to view the problem more generally, we will refer to the receiver end of the channel as the channel **output**, Y, and the transmitter end of the channel as the channel **input**, X. So, the unit sample response is Y, given $X = \delta$

Notice that we can express *any* input signal X in terms of a time-shifted addition of unit samples, by writing each of its elements $x[n]$ as follows:

$$x[n] = x[0]\delta[n] + x[1]\delta[n-1] + \ldots + x[n]\delta[0]. \tag{1.2}$$

Figure 1-3 shows why X can be deconstructed in this fashion.

Notice that in Equation 1.2, only one term in the sum is non-zero for any given value of n. From the definition of the unit sample response, H, and from the decomposition of X in

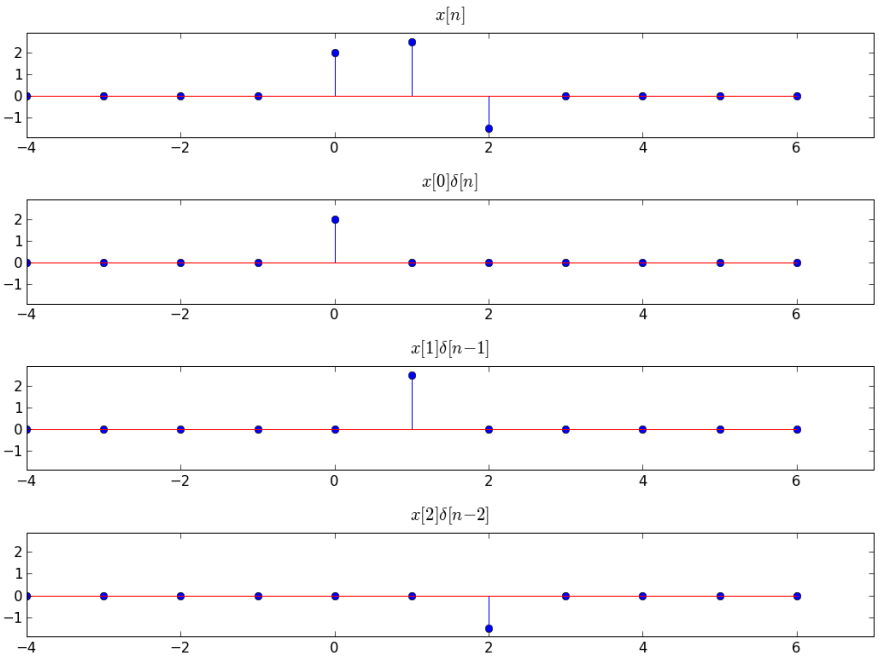

Figure 1-3: A diagrammatic representation of representing an input signal X as a sum of shifted weighted unit samples.

to a weighted sum of shifted unit samples, we can write the output, Y, as follows:

$$y[n] = x[0]h[n] + x[1]h[n-1] + x[2]h[n-2] + \ldots x[i]h[n-i] + \ldots + x[n]h[0]. \tag{1.3}$$

Notice how both linearity and time-invariance are used in Equation 1.3, as shifted and scaled unit sample responses are being summed to compute Y, given shifted and scaled unit samples represent X.

Equation 1.3 has a special name: it is called a **convolution sum**, or just a convolution. It is a remarkable result: it says that *the unit sample response completely characterizes a noise-free LTI channel*. (The previous sentence is worth re-reading!)

A more compact way to represent Equation 1.3 is in summation form,

$$y[n] = \sum_m x[m]h[n-m]. \tag{1.4}$$

The astute reader will note that we have not used causality anywhere in our discussion. In fact, Equation 1.4 is written sloppily, because we haven't paid attention to the range of values of m in the summation. A more precise expression is:

$$y[n] = \sum_{m=0}^{m=n} x[m]h[n-m]. \tag{1.5}$$

In this more precise expression, we have exploited causality by terminating the summation

when $m = n$. If we assume that $h[i] = 0$ for all values $i < 0$, then $h[n - m] = 0$ for $m > n$ and we can terminate the summation at $m = n$. But why is $h[i] = 0$ for $i < 0$? This is due to causality, because in a causal system, the output due to a unit sample input cannot before before the unit sample starts, at index 0.

To summarize:

> *When we send an input X through an LTI channel, the output $Y = H * X$, where $*$ is the convolution operator.*

The convolution sum *fully captures* what a noise-free LTI channel does to any input signal: if you know what happens to a unit sample (or unit step) input, then you can compute what happens for *any* input.

■ 1.4 Deconvolution

We're now ready to understand how to undo all the distorting effects of transmitting voltage samples through a wire or channel, provided the wire or channel is linear and time-invariant. We now know that sending a sequence of voltage samples through a wire or channel is exactly equivalent to convolving the voltage sample sequence with the wire or channel's unit sample response, H. Finite rise and fall times, slow settling or ringing, and inter-symbol interference are all be the result of convolving the voltage sample sequence generated by a link's transmitter with H. And, if we can undo the convolution, we can eliminate all these effects at once. Deconvolution, as the name implies, does just that.

To apply deconvolution in practice, we will of course have to know what the channel's unit sample response. No one is going to tell us H, and certainly H will be different for different channels, In fact, H may even change slowly with time for a given channel (think about walking around while talking on your cell phone), though formally that violates our time-invariance assumption. So, we will assume that at least for the duration of a particular transmission, the channel is time-invariant. In practice, the receiver will have to learn what H is, by having the transmitter periodically send known test data, from which the receiver can estimate H. Though often not so practical a strategy, let us assume that H is determined by inputting a unit sample or a unit step in to the channel, as the output from either input can be used, in principle, to estimate H.

Given H, and a sequence Y of output samples, how does the receiver construct what was sent? To solve this problem, note that we would like to solve the following problem. Find W, an estimate of X, such that

$$H * W = Y. \tag{1.6}$$

Recall that we are assuming that the receiver knows H and Y, but NOT X.

To solve Equation 1.6, we can rewrite Equation 1.5 with $w[n]$ (our estimate of $x[n]$) substituting for $x[n]$, and with some reordering of the summation indices, as

$$y[n] = \sum_{m=0}^{m=n} w[n - m]h[m]. \tag{1.7}$$

Equation 1.7 is equivalent to Equation 1.5 because all we did was to shift the indices around

(put another way, convolving two functions is commutative). Given this reordering of Equation 1.5, note that the causality condition is now captured in the $m = 0$ lower-limit in the summation.

To develop an algorithm to solve Equation 1.7, we need to make one more assumption, one that is reasonably accurately satisfied by typical communication channels: we will assume that the unit sample response, H, goes to zero after a finite number of samples. That is, we will require that there is some K for which $h[n] = 0, \forall n > K$. Given this K-length H, we can write

$$y[n] = h[0]w[n] + h[1]w[n-1] + h[2]w[n-2] + \ldots h[K]w[n-K]. \tag{1.8}$$

Equation 1.8 is an example of a **difference equation**. There are several ways of solving such equations, but this particular form is easy to solve using iterative substitution (aka "plug and chug"), and we can use that solution to write a simple iterative algorithm (you will do that in the lab for both synthetic and the IR channel).

With a little algebra, we can write $w[n]$ in terms of the known H, the received value $y[n]$, and the previously calculated W values, $w[0], \ldots, w[n-1]$,

$$w[n] = \frac{y[n] - (h[1]w[n-1] + h[2]w[n-2] + \ldots + h[K]w[n-K])}{h[0]}. \tag{1.9}$$

The careful reader will note that this equation blows up when $h[0] = 0$. In fact, one must take care in practice to handle the case when the leading values of H are all zero or very close to zero. A little thought will lead one to the conclusion that in a working algorithm, one must extract any leading zeros from H, and also suitably shift Y, before applying Equation 1.9.

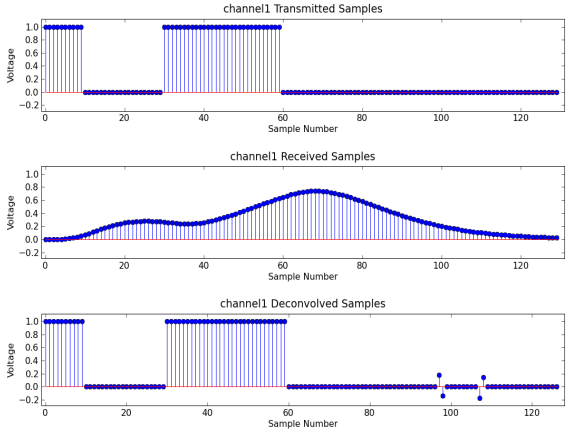

Figure 1-4: Deconvolution produces a fairly accurate reconstruction of the input when there's no noise.

■ 1.4.1 Evidence

How well can we undo the effects of ISI? In the absence of noise, it works nearly perfectly. Figure 1-4 shows an example, where the reconstruction of the input is essentially

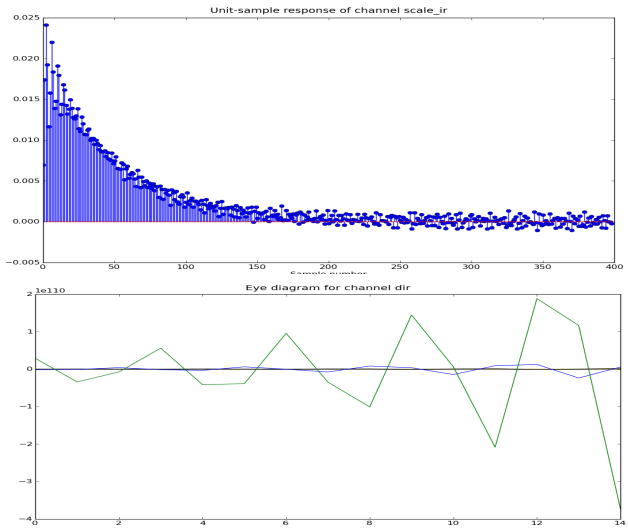

Figure 1-5: Deconvolution alone doesn't work if the noise is high—note that the scale on the y-axis of the eye diagram is "off the charts" (10^{110})!

completely correct. The only glitches appear around $n = 100$, which correspond to our truncating $h[n]$ to 0 for $n > 100$ or so.

Unfortunately, in the presence of noise, the performance is truly awful, as shown in Figure 1-5. The next two lectures will address this problem and develop techniques to understand and cope with noise.

■ 1.5 Some Observations about Unit Sample and Unit Step Responses

For a causal LTI (CLTI) system ($h[n] = 0, n < 0$), the unit step response is the cumulative sum of the unit sample response:

$$s[n] = \sum_{m=0}^{m=n} h[m], \tag{1.10}$$

where $s[n]$ denotes the step response at sample n. To derive this cumulative sum result, let the unit step be the input to a CLTI system, $x[n] = u[n]$. Then the convolution formula in (1.7), with w substituting for the input signal x, becomes

$$y[n] = \sum_{m=0}^{m=n} u[n-m]h[m] = \sum_{m=0}^{m=n} h[m], \tag{1.11}$$

where the sum at the extreme right follows easily by noting that $u[n-m] = 1$ for $0 \leq m \leq n$.

There is a reverse relation as well—the unit sample response for a CLTI system is equal to the difference between the unit step response and a one sample delayed unit step re-

sponse.

$$h[n] = s[n] - s[n-1]. \tag{1.12}$$

We can use the cumulative sum result to derive (1.12),

$$s[n] - s[n-1] = \left(\sum_{m=0}^{m=n} h[m] - \sum_{m=0}^{m=n-1} h[m] \right) \quad = \quad h[n], \tag{1.13}$$

where the result on the far right follows because all but the n^{th} term in in the subtracted sums cancel.

We can use the cumulative-sum and delay-difference results to make a number of observations about a CLTI system's step response given its unit sample response.

1. From the cumulative-sum result, it follows directly that is $h[n] \geq 0$ for all n, then the step response increases monotonically and never rings. From the delay-difference result, it also follows that if the step response does ring, then there *must* be values of n for which $h[n] < 0$.

2. if $h[n] > \varepsilon$ for all n, where ε is a positive number, then the unit step response keeps increasing towards infinity, $\lim_{n\to\infty} s[n] = \infty$.

3. If $h[n] = 0$ for $n \geq K$, then the step response is completely settled after K samples. That is, $s[n] = s[K]$ for $n > K$.

4. Suppose

$$s[n] = 1 - \alpha^{n+1} \tag{1.14}$$

for $n \geq 0$ and for some arbitrary value of α, $0 \leq \alpha \leq 1$. For example, suppose $\alpha = \frac{1}{2}$. Then $s[0] = \frac{1}{2}$, $s[1] = \frac{3}{4}$, $s[2] = \frac{7}{8}$, and $\lim_{n\to\infty} s[n] = 1$. Notice that the step response rises rapid from zero, but then approaches one in ever decreasing steps. The associated unit sample response is $h[n] = (1-\alpha)\alpha^n$ for $n \geq 0$. For the same example $\alpha = \frac{1}{2}$, $h[0] = \frac{1}{2}$, $h[1] = \frac{1}{4}$, $h[2] = \frac{1}{8}$, etc. Notice that $h[n]$ has its largest value at $n = 0$, and then monotonically decays. Completely consistent with a unit step response that is making progressively smaller steps towards its steady state.

5. Suppose $h[n] = \frac{1}{N}$ for $0 \leq n \leq (N-1)$ (a N sample long boxcar of height $\frac{1}{N}$). Then the unit sample response will form a ramp that increases from zero to one in a straight line, and then flattens out (such a function is called a saturated rample). The formula for this saturated rampd is then $s[n] = \frac{n+1}{N}$ for $0 \leq n \leq (N-1)$, and $s[n] = 1$ for $n \geq N$. Notice that a boxcar unit sample response produces a saturated ramp for a unit step response.

■ 1.6 Dealing with Non-Ideal Channels: Non-LTI → LTI

Many channels are not linear. For example, consider a channel whose output $y[n]$ is given by $y[n] = 1 + \frac{x[n]}{2}$, where $x[n]$ is the input to the channel. This system is not linear; it is formally known as an *affine system*. To see why this system is not linear, let's set $x_1[n] = 1$ for all $n > 0$. Then $y_1[n] = \frac{3}{2}$ for $n > 0$. If we create a new input by scaling X_1, $x_2[n] = 3x_1[n]$, then $y_2[n] = \frac{5}{2} \neq 3y_1[n]$, violating linearity.

But we can convert this channel in to a linear one. If we define $v[n] = y[n] - 1$, then $v[n] = \frac{x[n]}{2}$, and clearly the system that relates V to Y is linear. So what we can do given an output Y is to shift it by subtracting one, and then just work with the resulting signal V. Because V is the result of an LTI system operating on X, we can apply the techniques of this lecture to recover X, even though the actual physical channel itself is nonlinear. We will use such an approach in Lab 2 to make the nonlinear IR channel look like a linear system.

Obviously, channels could have much more complicated nonlinear relationships between inputs and outputs, but frequently there is a linear system that is a sufficiently good approximation of the original system. In these situations, we can apply tools like convolution and deconvolution successfully.