EE 123: Digital Signal Processing **Spring 2007**

Lecture 14 — March 1

Lecturer: Prof. Anant Sahai *Scribe: Matt Johnston*

14.1 Outline

- FFT Applied
- Discrete Time Processing of Continuous Time Signals
- OFDM (Orthogonal Frequency Division Multiple Access) Example
- (Return Midterms)

14.2 Review of Fast Fourier Transform

Last Time, we discussed how the FFT can compute a discrete fourier transform very quickly in $O(nlog(n))$ time, or exactly in $2nlog(n)$ time. In some Digital Signal Processors (DSP's), we can combine multiplications and additions into one operation, reducing the computation time from $2nlog(n)$ to $nlog(n)$. Therefore, the longer the length of the sequence being transformed, the more beneficial the use of the FFT is.

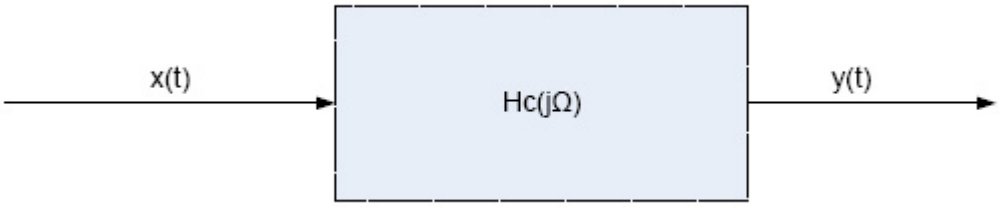
So far, we have analyzed the connection between Discrete Time Convolution and Finite Time Convolution, and discussed the relationship as going from discrete time to finite time. In this lecture, we focus on making the connection between continuous and discrete time.

14.3 Discrete Time Processing of Continuous Signals

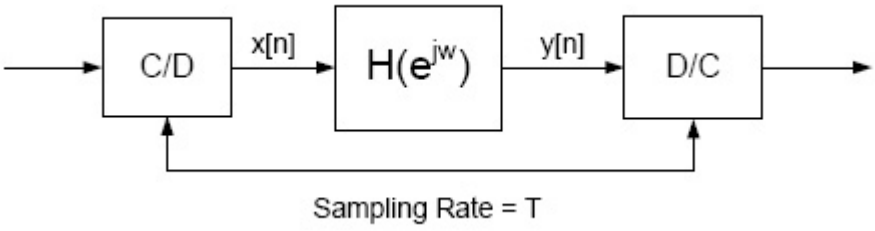
How does one go from continuous to discrete time? *Sampling*

14.3.1 Time Domain

Imagine we have a signal $x(t)$ and want to implement the following system:

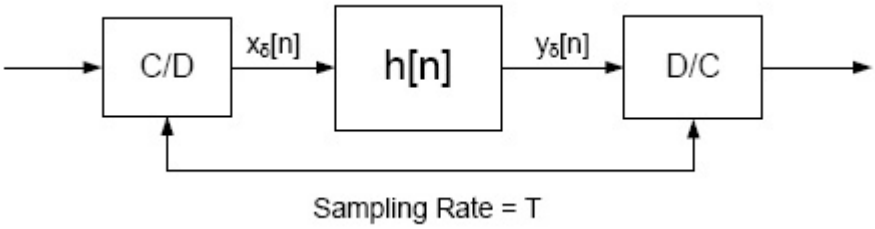


but we want to implement this system using Discrete Time processing. Therefore, the following system needs to be in the box above.



Now we want to calculate how the Ω in $H_c(j\Omega)$ and the ω in $H(e^{j\omega})$ are related, or how $h(t)$ is related to $h[n]$.

Lets assume there are no losses due to aliasing. To relate the impulse responses, we can start by sending an impulse through the system. However, it is difficult to picture what happens when an impulse is sent through a sampler. Therefore, we will look at this problem from a different angle. Assuming we have the system

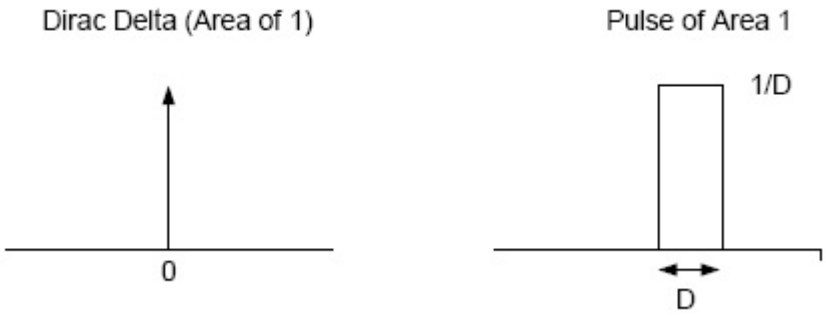


then

$$y_\delta[n] = h_c(nT). \tag{14.1}$$

From here, we can try to figure out $x_\delta[n]$.

Intuitively, we want to have $x[n] = 0 \ \forall n \neq 0$. If we assume this, then we only need to find the value of $x[n]$ at $n = 0$. We approach this problem by looking at the model of the dirac delta. A dirac delta is nothing but the limit of narrower and narrower pulses as seen below.



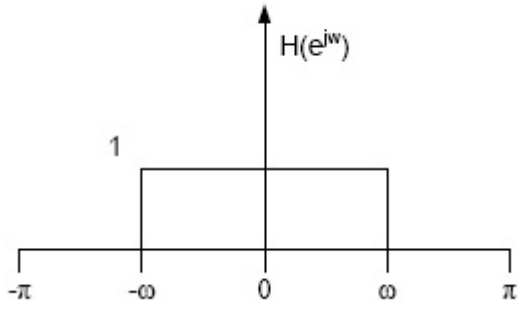
As D approaches zero, then the height of the pulse approaches infinity, and the pulse becomes a dirac delta. As long as $D < T$, the pulse will only be sampled once. Therefore, we can see that the first point that this happens at is $D = T$. Therefore, $x[n] = \{ \begin{matrix} \frac{1}{T} & if n = 0 \\ 0 & if n \neq 0 \end{matrix}$

From this, we can infer that

$$h[n] = T \times h(nT) \tag{14.2}$$

14.3.2 Frequency Domain

Now we can look at the same problem of comparing the continuous and discrete time systems, but instead of looking at the impulse responses of each we can look at their frequency responses.
As an example, lets take $H(e^{j\omega})$ to be the frequency response of a LPF.



On this graph, the π and $-\pi$ of the DTFT correspond to a frequency of $\frac{1}{2T}$ and -$\frac{1}{2T}$ respectively and a value of Ω of $\frac{\pi}{T}$ and -$\frac{\pi}{T}$

Therefore, to do the translation from Ω to ω we just use

$$\Omega_c = \frac{\omega_c}{T} \tag{14.3}$$

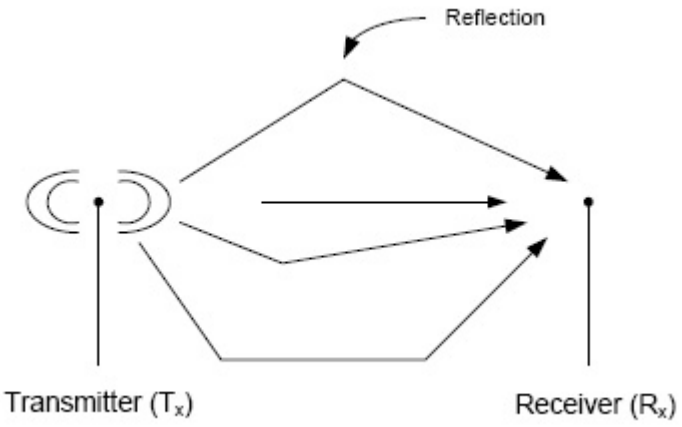
These relationships are very useful because there will be instances where we want to decrease the sampling rate, and in such scenarios we can practically read off what will happen to $h[n]$ and $H(e^{j\omega})$

14.4 OFDM Example

This example is used widely in digital communication, particularly in Wireless 802.11a and 802.11g standards.

14.4.1 Wireless Motivation

Signals are broadcasted from a wireless transmitter (T_x) and picked up by a receiver (R_x). However, as shown below, signals sent in directions not aimed at the receiver can still be picked up if the signals reflect of other environmental surfaces.



Every path from T_x to R_x has its own distance and attenuation. In many cases, we are interested in the reflected signals as much if not more than we are interested in the directly transmitted signal. For example, in many laser imaging applications the reflected signals are used to infer what type of objects are located in an environment and where they are located. We can idealize this wireless system in Continuous Time, by writing an equation for the impulse response of the system.

$$h_c(t) = \sum_{i=1}^{m} \alpha_i \cdot \delta(t - t_i) \tag{14.4}$$

When we want two transmit a signal, there are two different scenarios we must consider: transmission when we know the impulse response $h_c(t)$ and transmission when we don't.

14.4.2 Transmission with a known $h_c(t)$

Let us consider transmitting a signal $x(t)$. (Note: $x(t)$ should really be the message to be transmitted convolve with some carrier signal, but since this was covered in EE120, it will

be ignored here). We can write

$$x(t) = \sum_{i=-\infty}^{\infty} x[i]p(t - iT) \tag{14.5}$$

where the values $x[i]$ are complex signal values and the different $p(t)$ are pulses that are finite in time. We are more interested in having these pulses be finite in time, because they approximate real signals.

If the received signal, is $y(t) = (x * h_c)(t)$ and we sample the system at a period of T, then $y[n] = y(nT)$ and $h[n]$ is a discrete-time filter that gives the relationship of $x[i] \rightarrow y[i]$

If h_c is known, then h is known. Remember, our objective is to send values of x that we care about, and recover them. So can we just invert h since we know it? That will work as long as h has no zeros on the unit circle.

Now the real world system will use convolution of the as shown above to "compute" the output. However, our finite time filter will have to use circular convolution. So, is there a way we can force the real world to use circular convolution? We can if we make the input signal x periodic. Then the following steps can be taken to force circular convolution:

- Take the FFT of $y[1], y[2], ...y[n]$
- Take the FFT of $h[1], h[2], ...h[n]$
- Divide
- Take the inverse FFT to get back n numbers you care about.

However, there is a problem with this approach. If we make $x[t]$ periodic, then it will not be finite any more. Perhaps it only has to be "periodic enough." **Periodic Padding** is the process of padding your signals with periodic values, and is analogous to zero padding. For example, let us say x is of length 5 and h is of length 3. Then we are convolving

$$\begin{array}{cccccccccccc} - & - & - & - & x_0 & x_1 & x_2 & x_3 & x_4 & - & - & - & - \\ & & & & h_0 & h_1 & h_2 & & & & & & \end{array}$$

Lets try padding x with 3 digits on both sides, so are input becomes

$$x_2 \quad x_3 \quad x_4 \quad x_0 \quad x_1 \quad x_2 \quad x_3 \quad x_4 \quad x_1 \quad x_2 \quad x_3$$

When the convolution is carried out, the first two values of y are incorrect, and everything else is fine. This implies that we only needed to pad the sequence by two periodic values of x. Therefore, having a sequence

$x_3 \quad x_4 \quad x_0 \quad x_1 \quad x_2 \quad x_3 \quad x_4$

is enough to carry out the convolution. The two digits on the front of the sequence are referred to as a **cyclic prefix**. The cyclic prefix needs to be of a length one less then that of the FIR. A **cyclic postfix** is equally effective, but involves adding the extra periodic terms at the end of the sequence instead. If that were the case, then the sequence we would get from the convolution would correspond to the circular convolution of

$x_2 \quad x_3 \quad x_4 \quad x_0 \quad x_1$

So to avoid the circular shifting needed to recover the correct sequence, we will use the cyclic prefix. The prefix can be considered as useless information, because while necessary for the convolution, it does not carry any information.

So now how many operations does the receiver have to do?

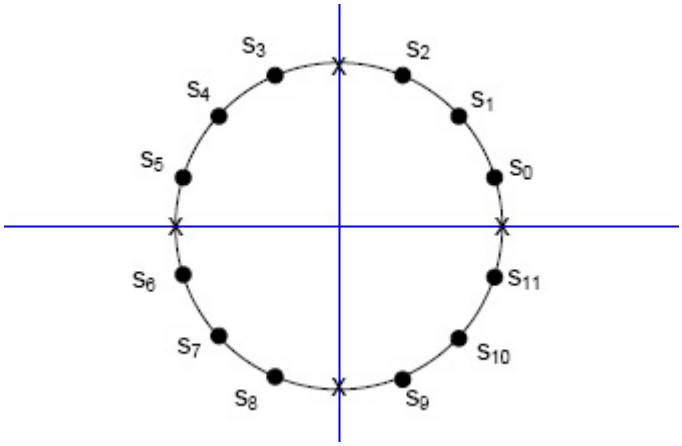
- Take the FFT of $y[1], y[2], ...y[n]$ $(2n \cdot log(n))$
- Take the FFT of $h[1], h[2], ...h[n]$ (*precomputed*)
- Divide (n)
- Take the inverse FFT to get back n numbers you care about. $(2n \cdot log(n))$

Perhaps we can cove some of this work to the transmitter. If the transmitter knows the impulse response, we can technically give it all of the work. However, we ideally want to split the work among the transmitter and the receiver. The transmitter has n numbers it cares about and wants to transmit. That means we have to construct a $n + k - 1$ length sequence (for $k =$ number of terms in the FIR filter). We can take an inverse FFT to get x, and then pad with zeros. Actually it doesn't matter what we pad with at some letter since it wont be seen as actual information. So for these purposes, we can pad it with anything.

14.4.3 Transmission with an unknown $h_c(t)$

When $h_c(t)$ is unknown, we can still work out this problem if the length of $h_c(t)$ known (say, when we know the longest path of a bouncing signal to the receiver from the first figure in this section). One option is to transmit an impulse, wait an certain amount of time (which we know since we know the length of the impulse) and by that point the receiver should have the impulse response. After that, we can proceed as in the previous section. The packet of n numbers that we want to send has to be padded with zeros to a length of $n + k - 1$. However, we are going have to pay a $3k - 3$ padding over head total, from sending the pulse, the cyclic prefix, and a third time for moving the signal over to the transmitter.

Another idea to note is that if an impulse response has four points, then knowing four points of a fourier response is enough to infer the impulse response. For example, suppose you have the points s_0 through s_11 like in the following picture.



Take the Inverse, 12-pt FFT to get a sequence in the finite time domain. Now pad, but instead of putting zeros at the end of the sequence, put one's at each eigenvector. This is equivalent to putting sampling points at the x's on the picture above. Now take the 16-point FFT, but at every 4 points a 4-point FFT will be taken, and from these points we can interpolate to recover h. After the divide step, the one's will remain and the other points will be corrected.

This is a complicated procedure, but it is how the 802.11a standard works (although that standard uses 64 samples instead of the 16 used in our example.

In summary, there are two tricks involved in this procedure. The first is to make nature obey your finite time model by adding a cyclic prefix, and the second is knowing the write place and write values to use as padding to recover the impulse response.

Midterm 1 was passed back at the end of class