EE20N: Structure and Interpretation of Systems and Signals **Fall 2013**

Lecture 16: October 24

Lecturer: Prof. Kannan Ramchandran *Scribe: TJ Tsai*

16.1 Agenda

- Meet the Fourier Transforms
- Clarifications
- Demo: Filtering Audio

16.2 Meet the Fourier Transforms

There seems to be a lot of confusion about the various transforms that we have worked with. We will try to clarify that confusion today. Figure 16.1 shows the high level picture. Let's look at each of these in turn.

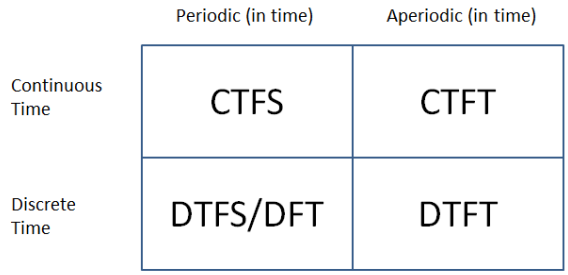

Figure 16.1: Relationship between various transforms.

16.3 CTFS: Continuous Time Fourier Series

The CTFS allows us to express a continuous time periodic signal in the frequency domain. The equations for the CTFS are:

$$x_p(t) = \sum_{k=-\infty}^{\infty} X[k] e^{ik\omega_0 t}$$

$$X[k] = \frac{1}{p} \int_{-T/2}^{T/2} x_p(t) e^{-ik\omega_0 t} dt$$

where p is the period of the time-domain signal and $\omega_0 = \frac{2\pi}{p}$. Because this signal is periodic, we only need to look at a finite window of time in order to understand this signal. Figure 16.2 shows a visualization of a continuous time periodic signal and its discrete CTFS coefficients.

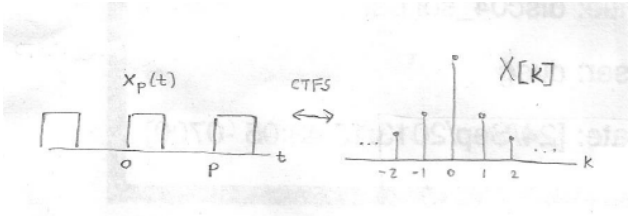

Figure 16.2: Visualization of CTFS transform pair.

16.4 DTFT: Discrete Time Fourier Transform

The DTFT allows us to express a discrete time aperiodic signal in the frequency domain. The equations for the DTFT are:

$$x[n] = \frac{1}{2\pi} \int_{-\pi}^{\pi} X_d(\omega) e^{i\omega n} d\omega$$

$$X_d(\omega) = \sum_{n=-\infty}^{\infty} x[n] e^{-i\omega n}$$

Because this signal is aperiodic, we need to look at its value over all time indices in order to compute its frequency representation. Figure 16.3 shows a visualization of a discrete time aperiodic signal and its frequency representation. Note $X_d(\omega)$ is continuous and periodic. This is essentially the flip scenario of the CTFS, where the time-domain signal was periodic and continuous and corresponding frequency-domain signal was discrete and aperiodic. In fact, CTFS and DTFT are time-frequency 'duals' where time and frequency have been swapped and $p = 2\pi$. Here we observe a few patterns. A signal that is periodic in time is discrete in frequency. A signal that is discrete in time is periodic in frequency. Combining these observations, we would expect a signal that is both periodic and discrete in time to be both periodic and discrete in frequency (see figure 16.4). This is preparing the way for the DFT.

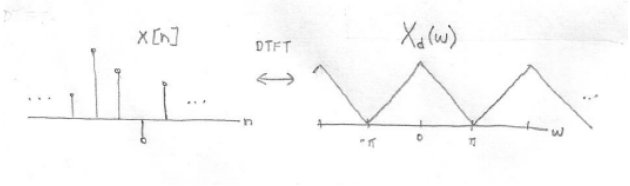

Figure 16.3: Visualization of DTFT transform pair.

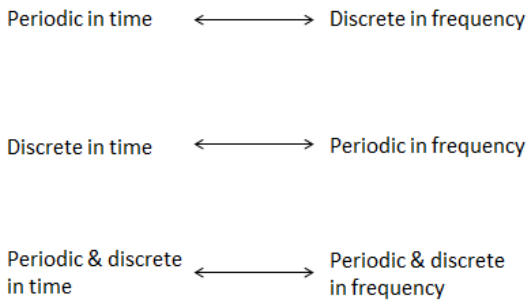

Figure 16.4: Patterns to notice about time-frequency pairs.

16.5 DTFS & DFT: Discrete Time Fourier Series & Discrete Fourier Transform

The DTFS is essentially the same thing as the DFT. The DTFS allows us to express a discrete time periodic signal in the frequency domain. The equations for the DTFS are:

$$x_p[n] = \frac{1}{p}\sum_{k=0}^{p-1} X_p[k]e^{ik\omega_0 n}$$

$$X_p[k] = \sum_{n=0}^{p-1} x_p[n]e^{-ik\omega_0 n}$$

where p is the period and $\omega_0 = \frac{2\pi}{p}$. The DFT is the mapping between one period of $x_p[n]$ and its frequency representation $X_p[k]$. There are two slight changes in notation that we use when using the DFT: (1) Drop the subscript p and (2) change p to N. So the equations for the DFT are:

$$x[n] = \frac{1}{N}\sum_{k=0}^{N-1} X[k]e^{ik\omega_0 n}, n = 0, 1, ..., N-1$$

$$X[k] = \sum_{n=0}^{N-1} x[n]e^{-ik\omega_0 n}, k = 0, 1, ..., N-1$$

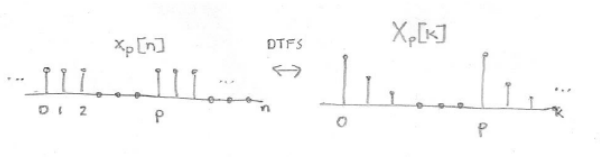

Figure 16.5: Visualization of DTFS transform pair.

Note that these two equations are computer friendly: they only require summations (rather than continuous integrals), and they only require dealing with a finite number of samples. Both the time domain representation $x[n]$ and the frequency domain representation $X[k]$ are discrete, so the

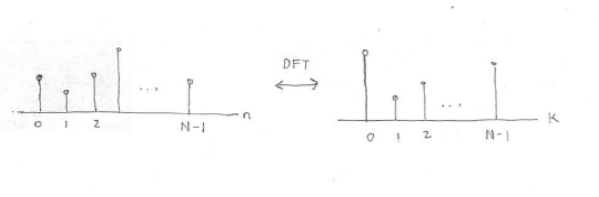

Figure 16.6: Visualization of DFT transform pair.

DFT is essentially taking one N-length vector containing the elements $x[0], x[1], ..., x[N-1]$ and converting it to another N-length vector containing the elements $X[0], X[1], ..., X[N-1]$. In fact, the DFT operation is simply a matrix multiplication. Here we simply introduce the definition of DFT. We will explore its properties more closely next time. Sample visualizations of DTFS and DFT are shown in figures 16.5 and 16.6.

16.6 CTFT: Continuous Time Fourier Transform

The CTFT allows us to express a continuous time aperiodic signal in the frequency domain. The equations for the CTFT are:

$$x(t) = \frac{1}{2\pi} \int_{-\infty}^{\infty} X(\omega) e^{i\omega t} d\omega$$

$$X(\omega) = \int_{-\infty}^{\infty} x(t) e^{-i\omega t} dt$$

We haven't covered this yet in this course, but we will soon. Stay tuned!

A quick note before moving on: for all 4 transforms discussed above, the equation that allows us to compute frequency domain representation is referred to as the analysis equation, and the equation that allows us to compute the time domain representation is referred to as the synthesis equation. It's good to know this terminology.

16.7 Clarifications

Now we'll take a short detour and address some points that have been confusing to many students.

16.7.1 Time and Frequency Domain

One area of confusion is the concept of time and frequency domain. Consider the analogous relationship between the rectangular and polar domain representations of complex numbers (see figure 16.7). The same complex number can be represented in two different ways. Why would we want to have two different representations of the same thing? We know from experience that some operations are much easier to do in one domain than the other. For example, if I asked you to add two complex numbers that were represented in polar coordinates, you would probably convert the two numbers to their corresponding rectangular forms, add them in the rectangular domain, and then convert back to polar form. This is because adding is much easier in the rectangular representation, while multiplying is much easier in the polar representation. The same relationship

holds between the time domain and the frequency domain: they are simply two different ways of representing the same signal. When we want to performs operations on signals, we will find that some operations are much easier to do in one domain instead of the other. And in these cases, we can use the various transforms to convert between the time and frequency domains.

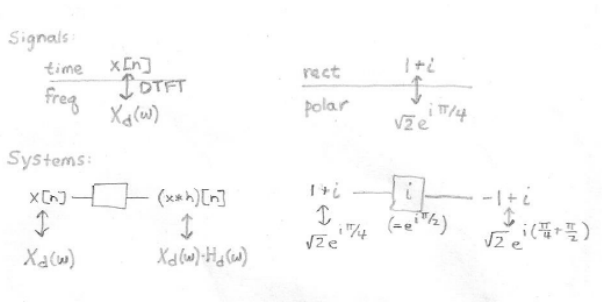

Figure 16.7: An analogy to help understand time and frequency domain.

Now let's consider how time and frequency domain representations help us to understand LTI systems. Again, let's return to our analogy of complex numbers. Consider a system that takes a complex number as an input and returns a complex number as an output. In this case, all it does is multiply by i. If we choose to represent our signals in the rectangular domain, we can describe what this system does by saying that it multiplies the input by i. If we choose to represent our signals in the polar domain, we can describe what this system does by saying that it adds a $\pi/2$ phase shift. Both of these descriptions are correct, and simply reflect which way we choose to represent the inputs and outputs. Likewise, we can describe what an LTI system does in two different ways. If we choose to represent our signals in the time domain, we can describe what it does by saying that it convolves the time-domain representation of the input with the impulse response of the system. If we choose to represent our signals in the frequency domain, we can describe what it does by saying that it multiplies the frequency-domain representation of the input with the frequency response of the system. Both of these descriptions are correct, and simply reflect which way we choose to represent the inputs and outputs.

16.7.2 Notation

These are the conventions that we will try to hold to for the remainder of this course:

- CTFT: $x(t) \leftrightarrow X(\omega)$
- CTFS: $x(t) \leftrightarrow X[k]$
- DTFT: $x[n] \leftrightarrow X_d(\omega)$
- DFT: $x[n] \leftrightarrow X[k]$

Note that parentheses denote continuous variables, while brackets denote discrete variables. The notation $H(e^{i\omega})$ denotes the frequency response of a discrete time LTI system.

16.8 Demo: Filtering audio

We have uploaded the demo code and files to bSpace, if you want to try out different filters (e.g. what happens if you apply a low-pass filter?)

16.8.1 DFT Basics

Let's start by looking at a basic discrete-time sine wave. **Question:** What is the frequency of this sine wave?

Answer: We can see that it completes one cycle in 8 samples, so it has a frequency of 1/8 cycles/sample. Since there are 2π radians/cycle, we have $f_d = 2\pi/8 = \pi/4$ rad/sample as our frequency. Now, we also know that

$$sin(\frac{\pi}{4}n) = \frac{e^{i\pi n/4} - e^{-i\pi n/4}}{2i}$$

Notice that this expression includes complex exponentials of two different frequencies: $\pi/4$ and $-\pi/4$. Since the Fourier transform aims to figure out which frequencies are present in a signal, we expect to see two separate impulses corresponding to exactly those frequencies. Indeed, if we take the DFT of the sine wave above, we get:

There are a few things that you should realize at this point:

1. When you take the DFT of a signal, the N frequencies corresponding to the N DFT coefficients range from 0 to 2π, not from $-\pi$ to π. You can see this from the formula for the DFT. However...

2. If you take the Fourier transform of a DT signal (DTFT or DFT), remember that the transformed signal is ALWAYS periodic in the frequency domain, with a period of 2π rad/sample. So even though the output of the DFT of a signal ranges from 0 to 2π according to our formula, you can equivalently represent the spectrum from $-\pi$ to π (or any interval of length 2π) by just rubber-stamping your DFT output across all frequencies, and then picking out your desired range of frequencies.

3. $\omega = \pi$ is the highest possible frequency for DT signals (in the range $[0, 2\pi]$). Similarly, 0 and 2π are LOW frequencies (0 is obviously low, and 2π is 2π rad/sample away from 0 rad/sample, so they are the same frequency). Therefore, if you're looking at the DFT magnitude plot for a signal and most of the energy is close to 0 and 2π (i.e. close to the ends), then the signal is mostly a low-frequency signal. On the other hand. if most of the energy is towards the middle of the DFT output (i.e. close to frequency $\omega = \pi$ rad/sample), then the signal is predominantly a high-frequency signal. So our sine wave $x[n]$ is a low-frequency signal.

16.8.2 Design challenge: Removing noise from audio

We begin by plotting the DFT of an audio clip (saved as 'clip.mp3' on bSpace). The spectrum (i.e. the frequency-domain representation of the signal) is plotted below: Is this signal predominantly low-frequency or high-frequency?

Now suppose the signal gets corrupted by a sine wave. You can listen to the variable 'noisy' in the MATLAB code. Now the spectrum looks like this:

Design Challenge: How do you remove the noise from this audio?

1. Do you prefer to work in time-domain or frequency domain?

   Answer: Frequency domain, because the noise is very easy to spot in the frequency domain, and it is concentrated, instead of being spread out all over the time domain.

2. What kinds of filters do you know?

   Answer: Low-pass, high-pass, band-pass, notch, comb

3. Which of these filters should you use and why?

   Answer: Notch, because it removes only two frequencies (ω and $-\omega$), while leaving the other frequencies untouched. Note, if you use a LP filter, you would have to remove everything between the 2 peaks as well as the peaks (remember that the high frequencies are in the middle of the plot!). This can really distort your signal, which is why you can't use a LP filter here.

4. The notch filter we saw in HW4 was continuous in frequency domain (call it $H_d(\omega)$). How do we convert that formula to a discrete-frequency filter that we can apply to our signal?

   Answer: We sample it! We already know that our filter must be as long as our signal, because we want to multiply them pointwise. Therefore, if there were N samples in the audio clip, we need N samples from the notch filter frequency response. In particular, we know exactly which frequencies are present in the DFT of our audio clip ($[0, \omega_o, 2\omega_o, \ldots, (N-1)\omega_o]$). So we can sample $H_d(\omega)$ at precisely those frequencies to get a discretized version of the notch filter. Plotting this frequency response gives:

   Remember that the frequency response here is discrete in frequency! It just looks continuous because ω_o is so small.

5. How do we actually apply this filter that we've built?

   Answer: We pointwise multiply our signal (in the frequency domain) by the discrete frequency response of the notch filter. Now you can play the filtered clip and hear that the buzzing is gone.

6. Is the filtered signal the same as the original signal?

   Answer: No. The filtered signal is missing the frequencies that our notch filter cut out, plus a little bit of the surrounding frequencies.

7. What if our noise is not just a sine wave? How can we filter, for instance, a ringing telephone? (you can listen to the clip in the matlab script)

   Answer: You could use cascaded notch filters. Each notch filter should remove two of the peaks from the noisy signal. Note that you probably can't use a comb filter here, because the teeth of a comb filter are evenly spaced apart. However, these frequencies are unevenly spaced. This makes sense because the phone ring doesn't have a single pitch, which means that it is not composed of a fundamental frequency plus the resulting harmonics.

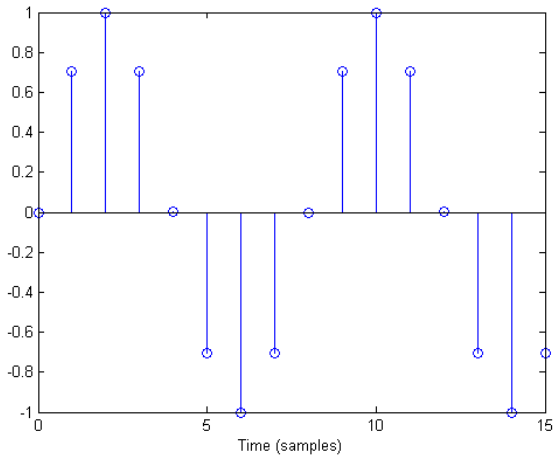

Figure 16.8: Sine wave $x[n]$.

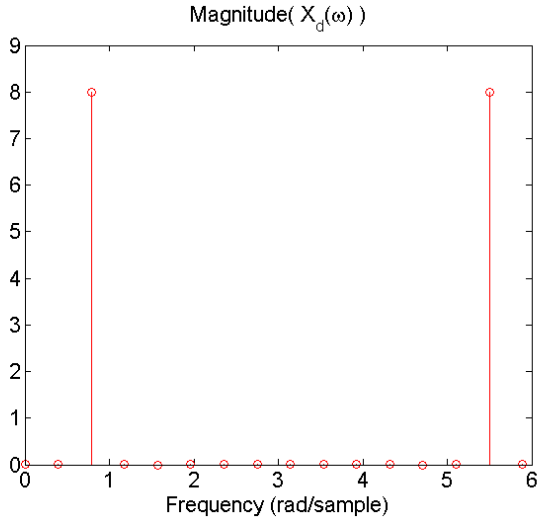

Figure 16.9: DFT of the sine wave $x[n]$.

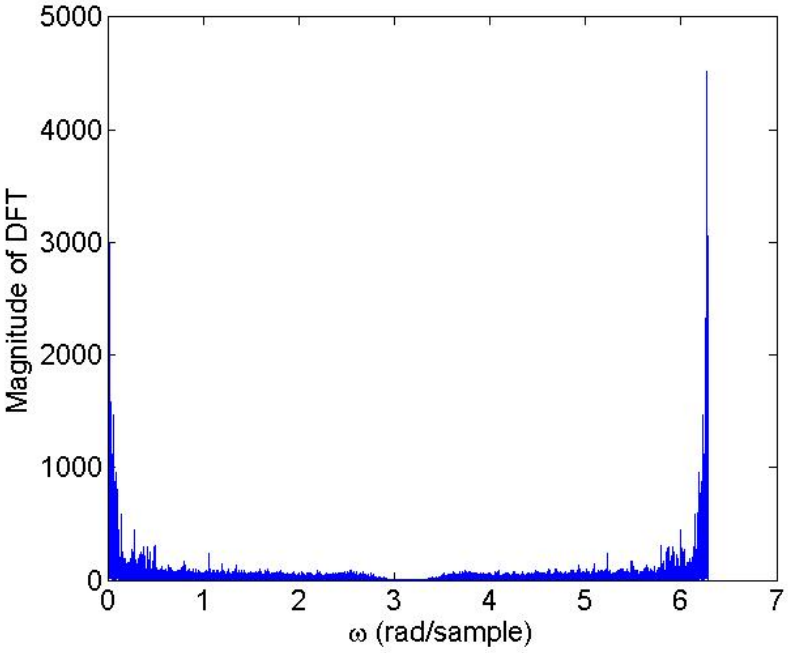

Figure 16.10: DFT of an audio clip ('Beautiful', by Snoop Dogg ft. Pharrell).

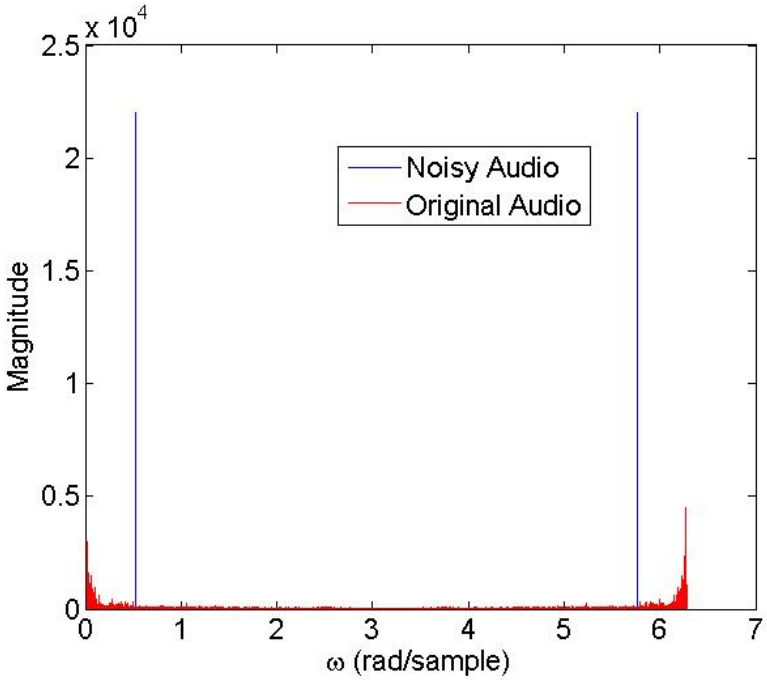

Figure 16.11: The noisy spectrum compared to the original spectrum. Notice that now we have the two peaks of a sine wave superimposed over our original signal.

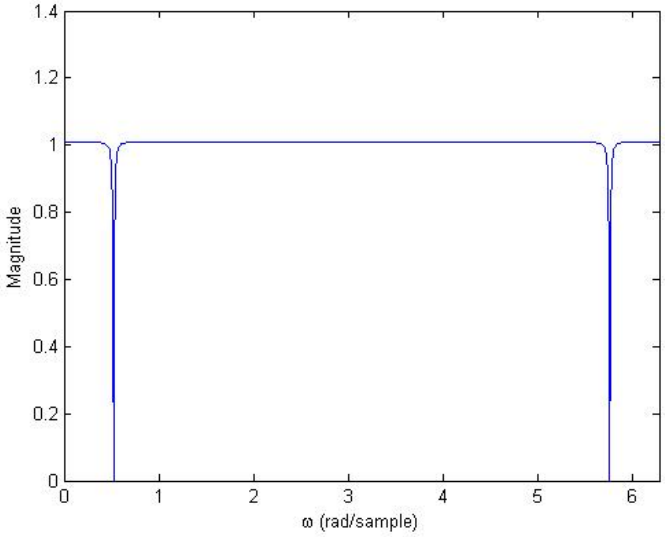

Figure 16.12: The spectrum (DFT) of our notch filter. We chose $\omega_c = \pi/6$ and $\alpha = 0.99$. The choice of α was pretty arbitrary, but the choice for ω_c was made so that ω_c and $-\omega_c$ line up with the frequencies of the magnitude peaks of the sine wave.

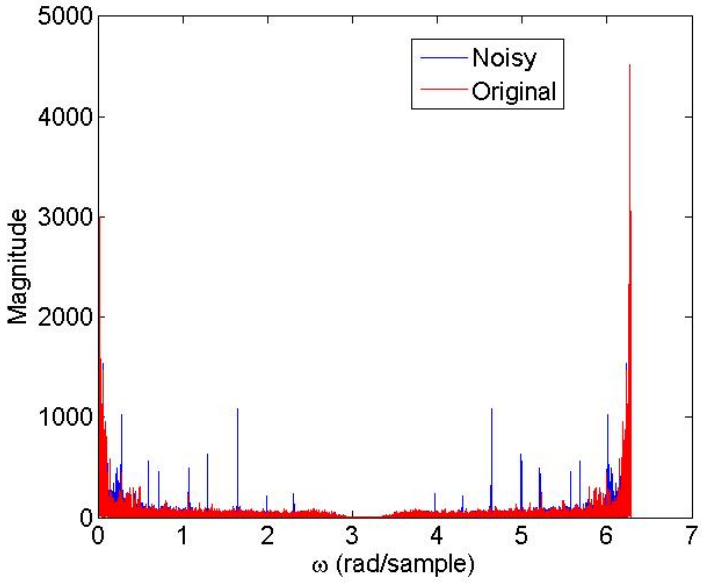

Figure 16.13: The spectrum (DFT) of the audio clip with a telephone ringing at the same time.