ECEN 455 - Digital Communications

Lab 1 - Signals and Spectra

Fall 2015

Introduction

Much of digital communications at the physical layer involves processing of various signals. In this first lab, we will review some concepts of signals from ECEN 314. You will view signals in the time domain and in the frequency domain and build some tools that will be useful in later labs in the course.

Discrete Time Representations of Continuous Time Signals

Consider a continuous time signal $x(t)$. This may be something as simple as a sine wave, or it could be a segment of speech, or the recording of an alien transmission received from a distant galaxy. When we store (or process) this signal on a computer, it must be done in a digital fashion since our computers ultimately deal with bits (1's and 0's). The first step in this process is to convert the continuous time signal to a discrete time signal, $x[n]$. The relationship between the discrete time signal and the continuous time signal is a simple matter of sampling

$$x[n] = x(nT_s), \tag{1}$$

where T_s is the sampling interval. Its inverse, $f_s = 1/T_s$ is the sampling frequency. Note we are using (curved) parentheses for continuous time and (square) brackets for discrete time. So $x[n]$ is nothing more than the nth sample of $x(t)$. If the sampling rate is sufficently high, we could easily interpolate between the samples stored in $x[n]$ to form a very close representation of the original continuous time signal. Recall Nyquist's sampling theorem states that as long as $f_s \geq 2W$, where W is the bandwidth of the signal,

then it is possible to perfectly reconstruct $x(t)$ from its samples. So the fact that our digital computers require us to represent our continuous time signals in a discrete time manner does not impose any real limitation. We can always go back and forth between $x(t)$ and $x[n]$.

Since our labs are computer based, all of the signals we deal with will be in discrete time. We will consider a sampled version of a continuous time signal taken at a very high sampling rate to be a very good approximation to the original analog signal. We can then change the sampling rate of the signal to obtain various discrete time versions of the signal. The process of increasing or decresing the sampling rate of a discrete time signal is called upsampling or downsampling, respectively. You can think of the down sampling process whereby we go from a high sampling rate to a low sampling rate as akin to the process of converting from continuous time to discrete time. Similarly the process of upsampling whereby we go from a lower sampling rate to a higher one can be thought of as reconstructing the original continuous time signal from a sampled version.

The Fourier Transform (Spectrum) of a Signal

It is often useful to look at the frequency content of a signal. Recall that for a continuous time signal, $x(t)$, its Fourier transform is given by

$$X(f) = \int_{-\infty}^{\infty} x(t)e^{-j2\pi ft}dt. \tag{2}$$

Here the frequency variable, f, is specified in units of Hz (you may be used to seeing this relationship in terms of ω in radians/sec). The inverse relationship is

$$x(t) = \int_{-\infty}^{\infty} X(f)e^{j2\pi ft}dtf. \tag{3}$$

If we wanted to express the continuous time Fourier transform (CTFT) in terms of the discrete time signal $x[n]$, we could do so by approximating the integral as a Riemann sum,

$$X(f) \approx T_s \sum_n x(nT_s)e^{-j2\pi fnT_s} = T_s \sum_n x[n]e^{-j2\pi fnT_s}. \tag{4}$$

Ultimately we will have to store the Fourier transform in digital form as well, so we will need to represent frequency in discrete increments. Let

$X[k] = X(kf_o)$ where f_o is the frequency increment. Then

$$X[k] = T_s \sum_n x[n] e^{-j2\pi knf_o T_s}. \tag{5}$$

Assuming that our original continuous time signal had a duration that lasted for N samples, then we can write the limits of the sum accordingly

$$X[k] = T_s \sum_{n=0}^{N-1} x[n] e^{-j2\pi knf_o T_s}. \tag{6}$$

MATLAB has a function called `FFT` which computes the discrete Fourier Transform as follows

$$X[k] = \sum_{n=1}^{N} x[n] \exp(-j2\pi(k-1)(n-1)/N). \tag{7}$$

Note the offset in the discrete indicies due to the fact that MATLAB does not allow you to start counting from 0 and also the missing factor of T_s. But armed with this knowledge, we can now use MATLAB's `FFT` function to compute and display CTFTs.

Task 1

Write a MATLAB function `[Xf,f]=CTFT(x,fs)` that will compute and display (plot) the CTFT of an input signal. The inputs should be a vector `x` containing the samples of the signal in the time domain and the sampling rate `fs`. The outputs should be a vector `Xf` containing samples of the CTFT and a vector `f` containing the frequency samples that correspond to the points in the frequency domain at which the CTFT have been sampled. The range of frequencies should be between $[-f_s/2, f_s/2)$. The function should also display a plot of the magnitude spectrum. The x-axis should be frequency in Hz (or kHz or MHz), and the y-axis should show the magnitude of the CTFT in dB, that is $20 \log_{10} |X(f)|$. Test your function on a few different signals that you can easily compute the CTFTs by hand. Report the results of your test verifying the proper operation of your function. You may find some of the following MATLAB commands useful: `fft`, `fftshift`, `abs`.

Task 2

- Use MATLAB's `audiorecorder` to record about 5 seconds worth of speech. Use a sampling rate of 24kHz, 16 bits per sample, and 1 channel (i.e., mono). Plot the recorded signal vs. time (in seconds) and play back your recording to make sure you have recorded your voice properly. You should type `help audiorecorder` from the MATLAB command line to learn about the suite of functions needed to record and play audio signals.

- Using the function you created in Task 1, compute and plot the magnitude spectrum of your recorded voice signal. From your plot, estimate the frequency content of your recording. That is, what range of frequencies does your voice display meaningful frequency content. At what frequency is your voice the strongest? Note that since everbody's voice is different, your answers to this should not be the same as anybody else's.

- Downsample the signal by a factor of 2. That is, lower the sampling rate to 12kHz. Play the downsampled signal. Can you hear any differences between the original and downsampled signal. Plot the downsampled signal as a function of time and also plot the magnitude spectrum. Comment of the differences (if any) you see in the time domain and in the frequency domain.

- Downsample the signal by a factor of 8. That is, lower the sampling rate to 3kHz. Play the downsampled signal. Can you hear any differences between the original and downsampled signal. Plot the downsampled signal as a function of time and also plot the magnitude spectrum. Comment of the differences (if any) you see in the time domain and in the frequency domain.