

## EE354 Spring 2011 Lab 3: Sampling Theorem Theory and Simulation

In this lab you will explore the properties of the sampling theorem from a theoretical and simulation perspective. You will use Matlab to create and view signals and the oscilloscope to display both the time-domain and frequency-domain signals. This lab requires a formal lab report per the guidelines posted on the course website. Students should work individually, however, only one lab report per group of two (2) students needs to be submitted.

### Pre-Lab Theory

#### Part 1A: Sinusoidal Signal

Consider the following audio signal:

$$s_1(t) = 0.5 \cos(2\pi(11.025 \text{ kHz})t)$$

The signal is sampled using the following three sampling frequencies:

- $f_s = 44.1 \text{ kHz}$
  - $f_s = 22.05 \text{ kHz}$
  - $f_s = 8820 \text{ Hz}$
- (a) For each sampling frequency, determine all frequencies that appear in the sampled spectrum that are less than 20 kHz and report them in the Table below.
- (b) For each sampling frequency, sketch a theoretical plot of the sampled spectrum for all frequencies between -20 kHz and +20 kHz.

<b>Sinusoid Sampled Spectrum</b>						
<b>n</b>	$f_s = 44.1 \text{ kHz}$		$f_s = 22.05 \text{ kHz}$		$f_s = 8.820 \text{ kHz}$	
	$nf_s + f_0$	$nf_s - f_0$	$nf_s + f_0$	$nf_s - f_0$	$nf_s + f_0$	$nf_s - f_0$




**Part 1B: Chirp Signal**

Consider the following audio signal:

$$s_2(t) = 0.5 \cos(2\pi(2500 + 200t)t) \quad 0 \leq t \leq 10 \text{ sec}$$

Note that this is known as a “chirp” signal. In this case our signal will start at a frequency of 2.5 kHz and increase over time up to a maximum frequency of 4.5 kHz. It will have a **flat spectrum** from 2.5 kHz up to 4.5 kHz.

The signal is sampled using the following three sampling frequencies:

- $f_s = 44.1 \text{ kHz}$
- $f_s = 22.05 \text{ kHz}$
- $f_s = 8820 \text{ Hz}$

- (a) For each sampling frequency, determine all frequencies (or range of frequencies) that appear in the sampled spectrum that are less than 20 kHz, and report them in the Table below.
- (b) For each sampling frequency, sketch the sampled spectrum for all frequencies between -20 kHz and +20 kHz. (NB: accurately show overlapping spectra where appropriate).

<b>Chirp Sampled Spectrum</b>						
<b>n</b>	$f_s = 44.1 \text{ kHz}$		$f_s = 22.05 \text{ kHz}$		$f_s = 8.820 \text{ kHz}$	
	$nf_s + f_0$	$nf_s - f_0$	$nf_s + f_0$	$nf_s - f_0$	$nf_s + f_0$	$nf_s - f_0$




Instructor Verification: \_\_\_\_\_

## Part 1: Simulation

### Part 1A: Sinusoidal Signal

For the single-tone audio signal  $s_1(t)$ , create a Matlab simulation to validate your predicted results for the three different sampling frequencies. **Don't forget** that, in Matlab, the sample frequency is used to create the time vector for your signal. Also, don't forget the effect of sampling...in the frequency domain, you will see replications of the original analog signal's frequency spectrum.

Plot (on a single figure using the subplot command) the time-domain and frequency spectrum associated with each sampling frequency. The time-domain signal should show several cycles of the sinusoid, and the spectrum should be plotted as log magnitude (dB scale) and show all shifted copies of the spectrum between -20 kHz and 20 kHz

**NOTE:** The provided Matlab code below uses the FFT (Fast Fourier Transform) routine to generate the power spectrum of a given signal—specifically the `spec_analysis.m` function available on the course website. The FFT only displays the spectral content of a signal that falls below half the sampling frequency. In order to see shifted copies of the original spectrum (as the sampling theorem predicts), the sampled signal has to be interpolated to a higher effective sampling frequency. The Matlab `upsample.m` function does this for us, simply by stuffing zeros in between “true” sample values. Additionally, the `spec_analysis.m` function provides power in Watts for a particular load impedance. To solve for normalized power, use a 1Ω impedance; for experiments performed in the lab, however, use a 50Ω impedance.

Ultimately, we want to be able to compare all three signals at the same (or similar) effective sampling frequency. To correctly generate and display all shifted copies of your spectrum, use code similar to the following snippet:

```
fs = #####;           % Establish our Sampling Frequency
delt = 1/fs;          % Time increment between samples
t = #####;           % Setup the time vector
fs_soundcard = #####; % Sound card operates at fs = 44.1 kHz
fs_mult = floor(fs_soundcard./fs); % Interpolation multiplier
fs_up = fs.*fs_mult; % Effective Sampling Frequency

y = cos(2*pi*fc*t); % Original Signal, Sampled at Original Rate
y2 = upsample(y,fs_mult); % Interpolated Signal - Zero Insertion Technique

[freq,power] = spec_analysis(y2,fs_up,impedance); % Generate the Spectrum
plot(freq,power) % Plot the results in dB!
```

**Note:** For all of your displays, the frequency spectrum will have power plotted in dB. To properly display the signal you are interested in (a 100 dB range is meaningless for these signals), you must adjust the axes to a reasonable set of bounds. To do that, run the following commands to “zoom” in on a section of your plots before you print them or save them. To set the minimum dB value displayed to -40dB, and the max to +1dB, and to make the frequency axis display all computed frequency content:

```
min_dB = -40;
max_dB = 1;
% sets the min and max dB value displayed in the plot
min_freq = #####;
max_freq = #####;
% sets the min and max frequency displayed in the plot...all freqs will be
% displayed, adjust as necessary to get a range of -20 to +20 kHz
axis([min_freq max_freq min_dB max_dB])
grid on
```

## Part 1B: Chirp Signal

The Matlab command to output a chirp signal is: `chirp(t, fstart, tstop, fstop)`. In this case, we want to sweep from 2.5 kHz to 4.5 kHz over a period of 10 seconds. The variable “t” is the time axis you set up in your code (make sure it extends for the entire 10 seconds, or else you will get an error!). Thus, the command would look like:

```
S2 = chirp(t, 2500, 10, 4500)
```

### Helpful Hints

An outline of your code should do something like the following:

1. Establish your sampling frequency ( $f_s$ ).
2. Select a start and stop time, and generate a time-axis (t).
3. Generate your original signal as well as the interpolated copy of your signal (y and y2).
4. Calculate the spectrum of the signal and then plot it with the time-domain response.

When finished, your results should look similar to Figure 1 below.

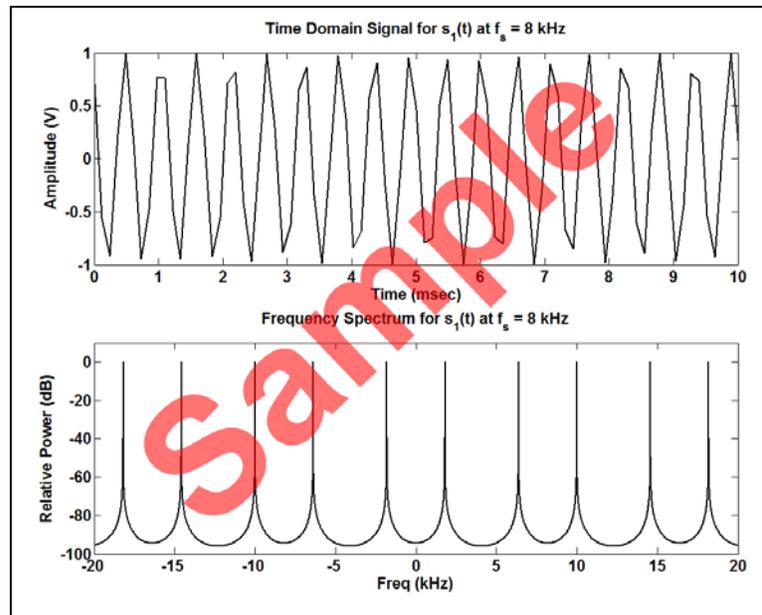


Figure 1: Example Time-Domain and Frequency-Domain Response for  $s_1(t)$  when sampled at 8 kHz.

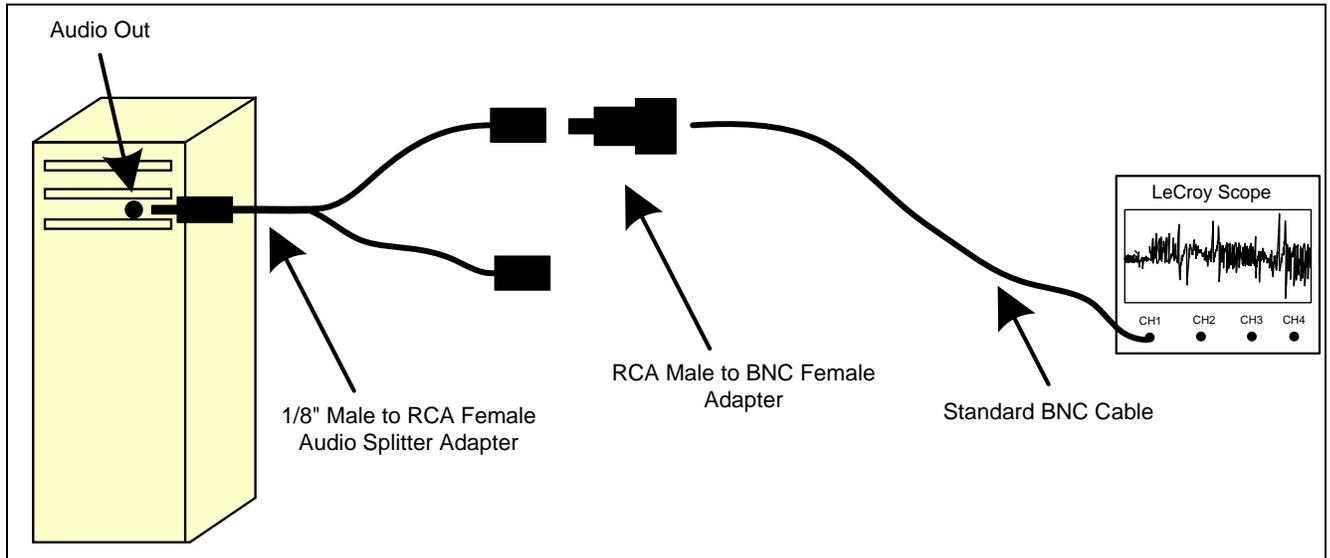
Show a meaningful amount of the time domain of each signal as well as the frequency spectrum from 0 – 20 kHz.

Instructor Verification: \_\_\_\_\_

## Part 2: Hardware Measurements of the Sampled Signal

### Part 2A: Single Tone

Next, use the oscilloscope to verify that all the predicted and simulated frequencies really exist in your sampled signal. Start by connecting the output of your PC's sound card to the Channel 1 input of your oscilloscope (you will require a set of adapters provided by the instructor), as shown below in the following diagram. **Note that:** the audio output jack of the PC at your bench may be at a different location than the one shown below.



**Figure 2: Illustration of the connection between the PC Audio Output and the LeCroy Oscilloscope.**

Configure the scope to record exactly 10,000 samples of the waveform. Set the sampling frequency to 50 kS/s, and configure the scope to display the frequency spectrum of the input signal.

After configuring the Oscilloscope, use Matlab to output the upsampled signal to your PC's sound card. The Matlab command to generate an audio output is `sound(<varname>, fs, Nbits)`. In this case, your upsampled waveform will be the <varname> output, `fs` will be your effective sampling frequency, and `Nbits` should be set to 16.

Observe the frequency spectrum on the scope and compare the results with theory/simulation. Use the cursors to place a vertical cursor on the FFT and verify that all predicted frequencies are actually present in the output.

Save a copy of both the time-domain signal and the frequency spectrum as recorded by the scope. Load the waveforms into Matlab, and generate a plot of the measured time-domain signal/measured frequency domain signal.

## **Part 2B: Chirp Signal**

Repeat Part 3A for the chirp audio signal  $s_2(t)$ . In order to see the spectrum on the oscilloscope, you will need to adjust the Math operation. Under the Math pop-up box, select Dual Function. Make sure the following options are enabled:

- Operator 1: FFT
- Operator 2: Roof
- Summary: Roof(FFT(C1))

Play the wave file as before. Use the **Clear Sweeps** button on the front panel of the scope to clear the screen in between observing waveforms. Save a copy of both the time-domain signal and the frequency spectrum as recorded by the scope. Load the waveforms into Matlab, and generate a plot of the measured time-domain signal/measured frequency domain signal.

**Include each of the six plots (three each for the tone and chirp signal) for the measured time-domain/frequency-domain signals. Show a meaningful amount of the time domain of each signal as well as the frequency spectrum from 0 – 20 kHz.**