EE432: Digital Signal Processing Fall 2012

Project 2: Sampling & Quantization

Assigned: Thurs 8/30/2012 **Due: Thurs 9/06/2012**

This project is intended to give you an introduction to quantization and sampling. The effects of changing the number of bits used in quantization and changing sampling rates will be explored. You will induce aliasing and observe the effects. You will use a music clip in this lab, available on Google Docs/Drive. The processing you will perform in this lab is done off-line in MATLAB (not in real-time). This means that you will read a music clip into MATLAB in its entirety, do the processing, then play back the processed music, instead of processing it while it is being read in.

(Note: music clips used in the context of course projects are for educational purposes only, and should only be used for this course.)

I. Preliminaries

1. Voice signals typically include frequency content up to approximately what frequency? Based on your answer, how fast should a voice signal be sampled to minimize aliasing? What sampling rate is typically used for voice signals? If an 8-bit ADC is used to digitize a voice channel at this sampling rate, what is the resulting bit rate?

2. Music signals typically include frequency content up to approximately what frequency (using the upper range of human hearing as a bound)? Based on your answer, how fast should a music signal be sampled to minimize aliasing? What are the standard parameters to digitize a music signal to CD quality (number of bits/sample and sampling rate)? What is the resulting bit rate when digitizing a 2-channel (stereo) music signal using CD quality parameters?

II. Input/Output of Wav Files

3. Download a copy of the music clip. This is a .wav file, which MATLAB can read. The MATLAB functions of interest for this type of audio file are: *wavread*, *audioplayer*, *play*, and *wavwrite*.

   -The *wavread* function will read in a music file, and can return to you variables containing (1) the music signal, (2) its sampling frequency and (3) the number of bits/sample. There are several different ways to call *wavread*; in this course you should always use it to return at least these three variables. If the music clip you downloaded is a STEREO recording (left and right channels) *wavread* will return a music vector that is *m* x 2 in dimensions (where *m* is the number of samples). Type >> **help wavread (**or **doc wavread)** to learn how to use it.

   -The *play* method of an object created by the *audioplayer* function can play a music vector to the sound card (either mono or stereo signals). Realize that the correct sampling frequency must be known to play the music correctly. You should always pass the actual sampling frequency to *audioplayer*. Type >> **help audioplayer** to learn how to use it and >> **help play** to learn how to use the *play* method.

   -The *wavwrite* function will allow you to take a music vector, a sampling frequency, and a number of bits per sample and create a .wav file that can be played in most music players (e.g. Windows Media Player). Type >> **help wavwrite** to learn how to use it.

4. Use *wavread* to read your music clip into MATLAB. Use *wavplay* to play it. Record the sampling frequency and number of bits/sample for your music clip.

Sampling frequency: _____ Number of bits/sample: _____

5. Now do some experimenting with the music signal:

   -Play the music with the incorrect sampling frequency. Try both faster and slower than the original, but a warning: do not try it too slow, or else it may take an incredibly long time to complete. Try no slower than perhaps 0.5*correct sampling frequency.

   -Play the music backwards. MATLAB has two functions to flip vectors or matrices from left-to-right (*fliplr*) or up-to-down (*flipud*). Since the music is read in as 2 long columns, representing the left and right channels, they need to be flipped up/down.

III. Quantization

6. Write a MATLAB function that will normalize an input signal so its values fall into the range from -1V to +1V. One way to do this is by subtracting the signal's minimum value, dividing by its range, multiplying by 2, and then subtracting 1. Doing this independently for each channel of a *stereo* signal will in general change the relative amplitudes of the two channels, but this approach will be fine for our purposes here. Be sure to check for and handle the all-zeros input to avoid divide-by-zero errors.

   The function should be called *RangeNormalize432* (so the m-file should be called *RangeNormalize432.m*). The usage will be:

```
>> y = RangeNormalize432(x);
```

7. Write a MATLAB function that will quantize an input audio signal to a specified number of bits. The input signal is assumed to be audio, but this function should call your *RangeNormalize432* function to ensure the values lie in the standard audio range (-1V to +1V). Hint: The `floor` function may come in handy.

   The function should be called *QuantizeAudio432* (so the m-file should be called *QuantizeAudio432.m*). The usage will be:

```
>> y = QuantizeAudio432(x,nBits);
```

   In the line above, *x* is the input signal that you wish to quantize and *nBits* is the number of bits to use in quantizing the signal. This function should give a warning if *nBits* is NOT an integer > 0, and return the signal unchanged in this case.

7. Take your music file, extract a single channel, quantize it to 10 bits, then listen to it. Do you notice a difference? Now try 8 bits, then 5 bits, then 3 bits. What do you notice about the sound? What is the source of the corrupting signal?

   Play your 5-bit quantized music to the professor and have him initial here: _____

8. Plot your (normalized) original, one channel signal vs. time, using the `stairs` function rather than the `plot` function. The reason to do this is that the `plot` command will connect the samples (draw a straight line between each sample value), while `stairs` will give you a stair-step plot, which emphasizes the digital nature of the signal. After you plot the normalized original signal, type `hold on`, then plot the 3-bit quantized signal on top in a different color (again, use `stairs` instead of `plot`). Type `hold off` so no more curves will be plotted on this plot. Your x-axis is time and should be in units of seconds. Zoom in on the plot so that approximately 10 msec are displayed. Label the plot and turn in with the report. Comment on the differences you see between the original and the quantized version.

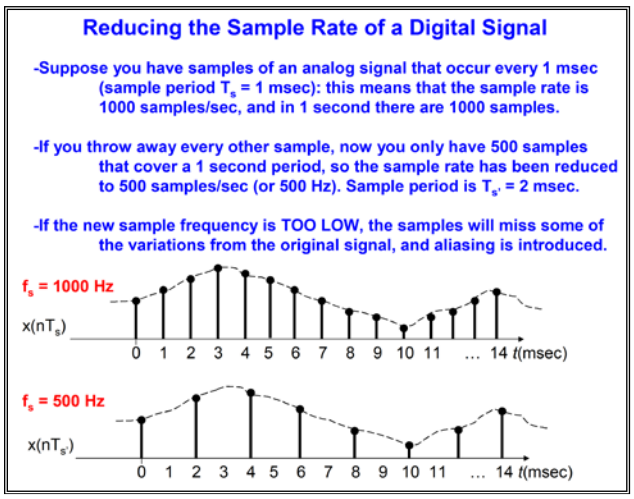
9. For the quantization you did in step 8, what is the max error in the quantization (note: the error signal is the difference between the original and the quantized version). Is the max quantization error consistent with the resolution of your quantizer?

10. Use `wavwrite` to write out a .wav file that is the original signal quantized to 3 bits, using your last name in the file name. After you write the new file, make sure it works by double-clicking on it in Windows. Upload this file to your Google Docs/Drive folder as part of your lab report.

IV. Time-Domain Aliasing

11. Induce aliasing into your stereo music signal. To do this, you need to effectively reduce the sampling rate. See the figure below to get the big picture of how to do that. Note that if the original sampling rate is 24,000 Hz and you keep every other sample, you've reduced the sampling rate to 12,000 Hz. If you keep only every third sample, the sampling rate is reduced to 8,000 Hz, etc. Do NOT use loops to achieve this; use the colon (:) operator instead. For your music clip, reduce the sampling rate by a factor of 10 and listen to the result using `play`. BE CAREFUL: For the new signal, (1) change the sampling rate that you pass to `audioplayer`, and (2) make sure that you apply the reduction in the number of samples to BOTH the left and right channels!

    Play your aliased music to the professor and have him initial here: _____

12. Now reduce the sampling frequency of the original file by a factor of 20. Listen to the result. Write out this aliased file using `wavwrite` and upload it to your Google Docs/Drive folder, ensuring that your last name is in the file name. Be sure to pass your new sampling frequency the `wavwrite` function! In your project report, describe the audible effects of the aliasing you've induced.

For the lab report: answer all of the questions above on a separate sheet of paper; turn in hard copies of your *RangeNormalize432* code and your *QuantizeAudio432* code; play your 5-bit quantized and aliased music signals for the professor; and upload your 3-bit quantized and aliased music files.