ECE431 Homework 4

Due Friday, September 28 at 3pm. Submit to ECE431 mobile file in WisCEL.

4.1. Oppenheim and Schafer 4.32.

4.2. Quantization Noise. Analog-to-digital converters sample continuous-time signals at uniformly spaced time intervals and quantize or round the amplitude of the sampled signal to a set of discrete levels in order to produce a value that can be stored in a computer. If a A/D converter has b bits of resolution, then it quantizes each sample to one of 2^b levels. Note that there are 2^b different numbers that can be represented using b bits - hence the number of levels. The difference between the true amplitude of a sample and the quantized amplitude may be modeled as a noise. Memory for storing the signal and A/D cost increase as b increases.

For example, CD quality audio is based on a 16-bit quantization because humans cannot audibly detect the difference between the "live" sound and sound sampled with 16 bits. Grayscale images are often quantized to 8 bits.

In this problem we will explore the characteristics of quantization noise using a music clip. We will assume that the amplitude of the music at each sample is continuous valued. This is a reasonable approximation because MATLAB employs double precision floating point numbers, which has much, much finer resolution than the quantization associated with the number of bits b used here to model an A/D converter. In MATLAB we will simulate quantization by applying the `round` command after scaling the signal to lie in the range from $-2^{b-1}+1$ to 2^{b-1}. The `sound` command assumes signals are scaled to the interval $[-1, 1]$, so to quantize a signal `song` that has amplitudes between -1 and 1, we may apply `round` as follows:

`quantsong = round(song*c)/c`

where `c` $= 2^{b-1} - 1$. That is, first we scale song by `c` so that the quantization levels are integers (signal range is $-2^{b-1}+1$ to $2^{b-1}-1$, then we round to quantize, and finally divide by `c` to restore the original amplitude range. For example, we may simulate quantization with $b = 6$ or (64 amplitude levels) with the command `quantsong6 = round(song*31)/31`. You can compute the quantization noise by subtracting the quantized signal from the original signal.

Load the data from file Bach44.mat. This contains `song` - the same Bach selection we've listened to before, but with the empty space at the start and endt removed.

a) Quantize `song` assuming $b = 10$, $b = 8$, and $b = 6$. Play the quantized and original signals using `sound(quantsong, fs)`. Can you hear the quantization noise? At what value of b?

b) Calculate the signal to quantization noise ratio (SQNR) in dB for $b = 10$, $b = 8$, and $b = 6$ by using `var` to compute the variance of the signal sf song and the variance of the quantization noise. Does your observed SQNR change by 6 dB/bit as the theory presented in class predicts?

For the remainder of this problem, assume $b = 6$.

c) Plot 500 values of the quantization noise from 5 different (well-separated) sections of the song. Is there a pattern evident?

d) Find the mean of the quantization noise and the maximum and minimum values. Is the mean approximately zero? How do the maximum and minimum values relate to b? Hint: use MATLAB commands `mean`, `max`, `min`

e) Plot a histogram of the quantization noise with 100 bins using `hist(quantnse,100)`. Is a uniform probability distribution a good model for the distribution of the noise amplitude?

f) Now calculate the quantization noise for a sinusoid $x[n] = \cos(2\pi n/32)$ for $n = 0, 1, 2, \ldots 9999$ and repeat parts c) and e).

If you are curious, I suggest listening to `song` after quantizing it to 1 bit, as follows: `signsong = sign(song);`. This forces positive values to +1 and negative values to -1. Can you still recognize the piece with only 1 bit of information?