Lab 4 – Sampling and Quantization

Objectives: In the last lab we studied sampling and reconstruction of *continuous-time* signals. In this lab we will

- study ADC and DAC of audio signals
- use Matlab to quantize *continuous-valued* signals, compute SQNR
- study effect of quantization on the quality of reconstructed signal.

4.1 ADC and DAC of audio signals (script)

You are given an audio file 'sp.wav'. Download and save it in your working folder so that the audio file and code are in same folder. Alternately you can record a short audio file (in .wav format) using any recording device/software in your computer and use it for this part.

>> Look up the audio file properties and note down its *bit rate*.

>> Look up documentation of the inbuilt matlab function `audioread()`. Use it to load the audio file in to your matlab workspace. What is the *sampling frequency* f_s of this audio signal?

>> From the length of the loaded signal and the *sampling frequency,* compute the duration of the audio signal in seconds.

>> From the observed *bit rate* and *sampling frequency*, compute how many bits the ADC must have used while quantizing/storing this signal. How many levels of quantization this ADC can perform?

>> Look up documentation of the inbuilt matlab function `sound()`. This function in combination with appropriate hardware in your computer performs DAC. Use it to listen to the audio file you have loaded in your workspace (ignore the 'nBits' input to this function).

>> Listen to the sound using lower sampling frequency than the true f_s (for example you can use $0.9\ f_s$, $0.8\ f_s$, $0.7\ f_s$, etc. while using `sound()` command). What do you notice?

>> Repeat the above using higher sampling frequency than the true f_s (for example you can use $1.2\ f_s$, $1.4\ f_s$, $1.6\ f_s$, etc. while using `sound()` command). What do you notice? (you may be reminded of certain fun apps children use in smartphones).

>> What property of Fourier transform can you use to explain your observations above.

4.2 Quantization (script + function)

>> Write a Matlab function `y = quant(x,L,a,b)` with inputs

- x -- input signal as a vector
- L -- Levels of quantization (positive integer)
- a & b -- the lower most and upper most levels for quantization.

The function should produce an output y, which is the uniformly quantized version of the input signal. For purpose of this function, assume that the input signal x itself is not quantized (though this is not true).

>> To implement the above function, uniformly divide the given range (a,b) into L equal sized intervals. The quantizer should map any value within the interval to the mid-point of that interval. As an example, if (a,b) = (-1,1) and L = 2, then for the i-th element of the input we have, if $x[i] \in [-1,0),\ y[i] = -0.5$ and if $x[i] \in [0,1],\ y[i] = 0.5$. This is only an example, the code should consider general values of (a,b) and any positive integer L.

>> Note that here we are only discretizing the levels of the output signal y. The encoding part which maps these levels to appropriately chosen binary code words is not considered.

>> Write a single script which does the following:

a) Consider the analog signal $x(t) = \sin(2\pi f_0 t)$ with $f_0 = 1KHz$. Sample a 1 second portion of this signal (in the time interval $t \in [0,1]$) at a sampling frequency of $F_s = 5KHz$ to obtain the discrete-time signal $x[n]$. Use your `quant()` function to obtain the quantised signal $x_q[n]$.

b) Create a figure with 3x1 subplot grid and plot the sampled signal and quantised signal in the first two subplots (use proper labelling). Use (a,b) = (-1,1) and repeat this for quantisation levels of $L = 2^B$ where number of bits used B = 1:8 in a for-loop. Note down your observations from figure.

c) Compute the quantization error $e_q[n] = x[n] - x_q[n]$. Plot this error in the remaining panel of the figure created above.

d) In another figure, plot a graph with number of bits (B, as used in part (b)) on X-axis and maximum absolute quantization error on the Y-axis for this signal and comment on your observations.

e) In another figure, plot a graph with number of bits (B, as used in part (b)) on X-axis and Signal to Quantization Noise Ratio (SQNR) on the Y-axis and comment on your observations. Experimentally measured SQNR is defined as the ratio of signal power to quantization noise power:

$$\text{SQNR} = \frac{\sum_n |x[n]|^2}{\sum_n \left|e_q^2[n]\right|^2}$$

f) Compare the experimentally measured SQNR with theoretical SQNR – reference reading is section 1.4.4, Eq. (1.4.32) of the textbook by Proakis & Manolakis.

4.3 Quantization of Audio signal (script)

For this section, use the audio file from part 4.1 above.

>> Load .wav file in Matlab. Quantize the signal using your `quant()` function and L = 8 levels. Listen to the original signal and the quantized signal using the `sound()` command. How does the sound quality of these two signals compare?

>> Perform quantization for different levels (L = 2,4,8,16,32,64,128,256) in a for-loop. Play the quantized signal in the for-loop with a pause of 2 seconds added after every call to the `sound()` command. Note your observations as levels increases and comment on quality of sound by hearing them.

>> What can be said about the frequency content of the quantized signals as the number of levels L is changed?