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 audio signals, their *bit rates* and *sampling frequency*
- use Matlab to quantize *continuous-valued* signals, compute SQNR
- study effect of quantization on the quality of reconstructed signal.

4.1 Audio signals

Download the 4 audio files given here. Save them in your working folder so that the audio files and code are in same folder. Write a **matlab script** for the following tasks.

>> 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 these audio signals?

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

>> From the observed *bit rate* and *sampling frequency*, compute how many bits the ADC must have used while quantizing/storing these signals. 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 files you have loaded in your workspace (ignore the 'nBits' input to this function).

>> For one of the files, 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?

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

>> Write down your answers as comments at the end of the code.

4.2 Quantization

Quantization involves discretization and encoding of samples of a signal. In this task we will take a closer look at discretization and how it effects the signal values. The encoding part which maps these levels to appropriately chosen binary code words is not considered here. For a discrete-time signal $x[n]$, quantization and quantization error are given by

$$x_q[n] = Q(x[n])$$
$$e_q[n] = x[n] - x_q[n].$$

The quantization function $Q()$ maps any real input to a point from discrete set of values. There are numerous quantization functions used in practice. We implement a non-uniform quantizer in the function below.

>> Write a Matlab function `y = quadratic_quant(x,B,a)` with inputs

- x – input signal as a vector
- B – number of bits used to decide quantization levels (positive integer)
- a – positive real number such that [-a,a) forms the range for quantization.

The function should produce an output y, which is the 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). We wish to implement a *quadratic non-uniform quantizer* as discussed below.

>> For the range $[0, a)$: To implement the above function, divide the interval $[0,1]$ into $L = 2^{B-1}$ equal sized intervals. Let $0 = r_0 < r_1 \ldots < r_{L-1} < r_L = 1$ be edge points of these intervals. Then the quantizer should map values in the interval $[ar_i^2, ar_{i+1}^2)$ to its mid-point. Repeat the process symmetrically for the range $[-a, 0)$. Make sure your quantizer has a total of 2^B levels in the interval $[-a, a)$.

As an example if a = 1 and B = 2, then the $2^B = 4$ quantizer intervals are $[-1, -0.25);\ [-0.25,0);\ [0,0.25);\ [0.25,1)$ and the quantization follows

if $x[i] \in [-1, -0.25),\ y[i] = -0.625$

if $x[i] \in [0, 0.25),\ y[i] = 0.125$

and so on. This is only an example; the code should consider general inputs 'a' and B. For inputs outside the interval $[-a, a)$ you should quantize them to the end points of your quantized set of values.

>> Write a **matlab script** which does the following tasks:

a) Consider the analog signal $x(t) = \sin(2\pi f_0 t)$ with $f_0 = 10\ Hz$. Sample a 1 second portion of this signal (in the time interval $t \in [0,1]$) at a sampling frequency of $F_s = 5\ KHz$ to obtain the discrete-time signal $x[n]$. Use the function above 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 = 1 and B = 4.
c) Compute and plot the quantization error in the remaining panel of the figure.

d) In a different **figure**, plot histogram of the quantization error using the matlab function `histogram()` with 15 bins (see documentation). Repeat for B = 3 and compare the histograms.

e) Repeat above processing for B = 1:8 (do not repeat the figures). In another **figure**, plot a graph with B on X-axis and maximum absolute quantization error (over the complete signal duration) on Y-axis and comment on your observations.

f) Experimentally measured SQNR is defined as the ratio of signal power to quantization noise power:

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

In another **figure**, plot a graph with B on X-axis and Signal to Quantization Noise Ratio (SQNR) on Y-axis and comment on your observations.

g) What can you say about accuracy of the above non-uniform quantizer in various regions of the interval $[-a, a)$? How would this compare with say a uniform quantizer which instead considers intervals of the form $[ar_i, ar_{i+1})$?

4.3 Quantization of Audio signals

For this section, use one of the audio file from part 4.1 and write a **matlab script**.

>> Load .wav file in Matlab. Quantize this signal using your quantization function with B = 3 and a = 1. 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 (B = 1:8) 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.

>> How does quantization affect the frequency content of the quantized signal compared to that of the input signal? How does B play a role in this?