Communication Systems
(ECE4572)
Fall 2013

Homework 4
Assigned Sept.26, due Oct.3

Objective: Review of sampling and quantization; proof-of-concept using Matlab.

Description: A clip of sound signal is described by

$$x(t) = \sum_{k=1}^{K} A_k cos(2\pi f_k t + \phi_k), t \in [0, T] \tag{1}$$

This signal has K components with different frequencies f_k, amplitudes A_k, and phases ϕ_k. The frequencies f_k span the desired hearing range in equal, pre-determined steps. The amplitudes A_k take values between 0 and 1, and the phases ϕ_k take values between 0 and 2π. In some cases, the amplitudes and the phases can be modeled as random variables, thus describing a set of sound signals.

The signal $x(t)$ is to be sampled at a rate f_s=8 kHz, then quantized using a uniform quantizer with $N = 2^{R_N}$ levels and a quantization step Δ.

Preparation:

- Review the sampling theorem. Express the signal $x(t)$ in terms of its samples $x(n/f_s)$ and the function $h(t) = \text{sinc}(\pi f_s t)$.
- Review the concept of quantization. Design a uniform quantizer for the signal $x(t)$, i.e. determine the quantization step Δ for a given number of quantization levels N. Note: The amplitude of the signal $x(t)$ is not uniformly distributed. Moreover, the maximum amplitude may vary from one sound clip to another. Hence, some care has to be taken to ensure that the signal is properly scaled before quantization. One possibility is to normalize the signal samples by their maximum value. This will ensure that the samples entering the quantizer have values between -1 and 1.

Proof-of-concept: Use the following signal parameters to implement the system in Matlab:

- total signal duration: T=2 s
- frequency components f_k: 100 Hz to 3400 Hz in equal steps of 30 Hz (K=111)
- signal amplitudes A_k and phases ϕ_k : anything you like[1]
- sampling frequency for mimicking continuous time: $F_s = 10 f_s$
- number of quantization levels: varying (R_N=3,4,5,6,7,8)

[1]You can also use the built-in random number generator (`rand`) to generate K independent amplitudes A_k, each uniformly distributed between 0 and 1, and K independent phases ϕ_k, each uniformly distributed between 0 and 2π.

Implementation and analysis:

1. Generate the signal $x(t)$, and save it. Note: To mimic continuous time in computer simulation, we chose a large sampling frequency, e.g. $F_s = 10f_s$. At this sampling frequency, the signal $x(t)$ will be represented by a vector $\mathbf{x}$ that will have as many elements as needed to cover the interval T in steps of $1/F_s$.

2. Sample the signal at f_s. The samples will be stored in a vector $\mathbf{x}_s$ whose length will be $I = F_s/f_s$ times shorter than the length of $\mathbf{x}$, but will cover the same time interval T.

3. Quantize the samples to N levels. The vector $\mathbf{q}$ of quantized samples will be of the same length as the vector $\mathbf{x}_s$.

4. Calculate the signal-to-quantization-noise ratio (SQNR) as the ratio of the power of $\mathbf{x}_s$ to the power of $\mathbf{x}_s - \mathbf{q}$. (The power of a vector is the average of the squares of its elements.) Express the SQNR in dB and store this value.

5. Repeat for different values of N. Plot the SQNR [dB] versus $R_N = log_2(N)$, the number of bits needed per quantization level. How does the SQNR increase with R_N? Is there a similarity with the theoretical value that we obtained in class for a uniformly distributed input signal? (If you want to see how the signal samples are distributed, you can plot their histogram – use the built-in function `hist`.)

6. For each value of N, calculate the bit rate R_b [bits/sec] needed to transmit the binary string that represents the signal.

Bonus:
Assume that the quantized values are encoded into binary (0/1) codewords of length R_N, and that these codewords are transmitted over a communication channel with no errors, i.e. that the receiver has a perfect copy of the vector $\mathbf{q}$. Recover the signal $\mathbf{x}$ from the quantized samples $\mathbf{q}$. Verify correct reconstruction by plotting a segment of the original signal and the corresponding recovered signal on the same graph. Do they look alike? Hint: Perform recovery according to the sampling theorem. Use the built-in `sinc` function. Note that you cannot perform ideal reconstruction since it would require a sinc of infinite duration. Instead, use a sinc truncated to $\pm L/f_s$ around the maximum point. For example, use $L = 8$. Once you have generated the truncated sinc, you can use the built in function `conv` to perform convolution, i.e. to filter the signal.

Report: Your typed report should contain:

- a cover page with your name
- a few paragraphs of text describing the problem, your work, and your conclusions
- figures that support your conclusions and to which you referred in text (figures must have captions; axes must be properly labeled and have adequate units, e.g. "time [s]")
- appendix containing your Matlab code.