PS #2 , Spring 2001
Signal Processing Using MATLAB, EECE-495
Instructor: Balu Santhanam
MATLAB Assignment
Date Assigned: 01/31/2001
Date Due: 02/07/2001

Background

The process of converting a discrete–time signal $x[n]$ with infinite precision to a discrete–time signal with finite–bit precision $x_q[n]$ is referred to as *quantization*. This will be necessary when the objective is to implement a DSP algorithm using a fixed point DSP processor. In mathematical terms this is denoted as :

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

We define the quantization error $e_q[n]$ as the error introduced in the quantization process via:

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

We will be studying two modes of quantiztion. The first mode is the *round-off* mode where the source sequence value is rounded to the nearest value in the output alphabet. The second mode is the *saturation* mode where the source alphabet is ceiled up to the nearest value in the output alphabet.

For the purposes of this exercise we will assume that our source $x[n]$ is sinusoidal with amplitude $X_{\max}$, i.e.,

$$x[n] \in [-X_{\max}, X_{\max}].$$

For a B bit quantizer with 1 bit allocated for the sign of the source, the stepsize of the desired uniform quantizer is given by:

$$\Delta = \frac{2X_{\max}}{2^{B+1}} = \frac{X_{\max}}{2^B}.$$

The average power of the sinusoidal input can then be computed via:

$$P_{\text{ave}}^x = \sigma_x^2 = \frac{X_{\max}^2}{2},$$

where σ_x^2 is the variance of the source $x[n]$. If on the otherhand the source produced a random signal $x[n]$ that was uniformly distributed, i.e, $x[n] \sim U([-X_{\max}, X_{\max}])$ then the average power of the source would be:

$$P_{\text{ave}}^x = \sigma_x^2 = \frac{X_{\max}^2}{3}.$$

In any case in the design of the quantizer we need to match the range of the quantizer data, i.e, $x[n] \in [-X_{\max}, X_{\max}]$ to the variance of the signal. Otherwise if we do get data that is not in this range then the quantizer will clip the signal.

It can be shown that in the case of the round-off and saturation options:

$$e_q^{(1)}[n] \sim U\left(\left[-\frac{\Delta}{2}, \frac{\Delta}{2}\right]\right) \quad , \quad e_q^{(2)}[n] \sim U\left([-\Delta, 0]\right).$$

The average power in the quantization error for either quantization type is therefore: given by:

$$P_{\text{ave}}^e = \sigma_e^2 = \frac{\Delta^2}{12} = \frac{X_{\max}^2}{2^{2B}12}.$$

The *signal to noise ratio* (SNR) of the output of the quantizer is defined via:

$$\text{SNR} = 10\log_{10}\left(\frac{\sigma_x^2}{\sigma_e^2}\right) \; dB.$$

For the specific case of the uniform quantizer this reduces to

$$\text{SNR} = 6.02B + 10.8 - 20\log_{10}\left(\frac{X_{\max}}{\sigma_x}\right) \; dB.$$

There is approximately 6 dB increase in the SNR of the quantized signal for every bit increase in resolution provided that we are in the linear range of operation of the quantizer and not clipping the signal, i.e, $\sigma_x \leq X_{\max}$.

Problem Outline

- Write a user–defined function `fxquant.m` that implements a B bit uniform quantizer with synopsis:

```
[x_q,alphab,SNR] = fxquant(x,B,opt)
B   : Bit resolution for unifrorm quantizer
x_q : Quantized signal with B bit resolution
alphab : Output alphabet
opt : quantization type option (1 or 2)
      1 --> round-off
      2--> saturation
```

  Specifically in the function the following must be implemented:

  1. A documentation header for the program describing the functionality of the program.
  2. Appropriate error checking to catch invalid input.
  3. Control flow commands to check for the option entered.
  4. Minimum number of for loops in the function.

- Check the function you wrote by using a input given by

```
>> x = cos(2*pi*[0:1:999]/1000); B = 7, B = 15;
```

  for both options.

- Plot the quantization error for both modes and in both the 8 bit and the 16 bit scenarios using the subplot environment.