Companding

Introduction

In this lab you will study the effect of companding on the quantization SNR of speech waveforms.

You will record a short sample of your own speech and then compute the quantization SNR with and without using μ-law companding.

By comparing the quantization SNR with and without the use of companding you will establish the advantage, if any, of using companding.

Quantized Speech Waveforms

Much of the information transmitted over data communications networks is speech. One way of transmitting a digitized audio waveform is known as Pulse Code Modulation (PCM).

The audio waveform is sampled at a fixed rate (typically 8 kHz for speech) and each sample is quantized using an analog-to-digital converter with a certain resolution, typically 8 bits for speech. The resulting 64 kb/s data stream is transmitted over a network. The receiver converts the samples back to a continuous analog voltage.

Speech Signal Levels

One of the characteristics of speech signals is that they have widely varying signal powers. In addition to variations due to how loudly someone speaks, the distance to the microphone and the gains and losses of various circuits, different sounds have different signal levels. For example, different phonemes (speech sounds) within a spoken phrase will have different signal powers.

See the Wikipedia article Help:IPA for English for a list of English phonemes.

Quantization Noise

Since the waveform is quantized to one of 256 levels (8 bits) the digitized waveform is not the same as the original waveform. The difference between the quantized and un-quantized waveforms is a type of distortion or noise called quantization noise.

The quantization noise power can be approximated by $q^2/12$ where q is the quantization step size.

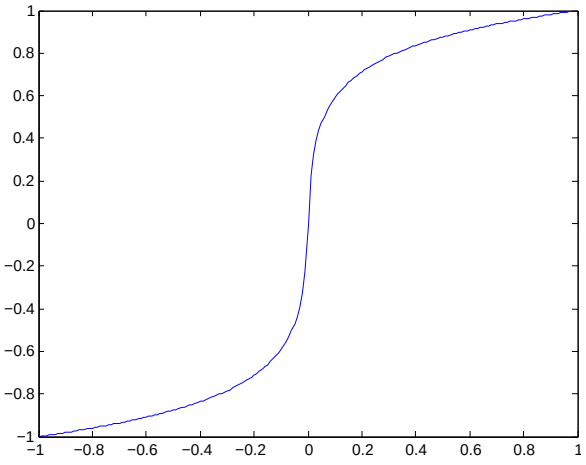
The gain before the A/D converter is set to avoid clipping the signal. Since the quantizer step size is the signal voltage range divided by the number of steps, the quantization noise will be determined only by the signal voltage.

For example if the A/D converter input levels are limited to $\pm V$ then $q = 2V/256$.

Effect of Companding

A compressor applies a non-linear function (logarithmic for $\mu-$law) to the audio waveform. This provides more gain to lower-level inputs and increases the average signal power while maintaining the same peak voltage level. This should increase the signal-to-quantization noise ratio, and thus the quality, of the quantized signal.

The μ-law compression function for $\mu = 256$ is shown below. The input level is along the x-axis and the corresponding output level along the y-axis.



Software

In this lab you will use the free audio editing program Audacity to record and select a segment of your

speech. You will use Matlab (or one of the free compatible programs) to apply the compression, expansion and quantization functions to the waveform and to compute the signal and quantization noise powers.

Companding and Quantization Functions

The μ-law compression formula is:

$$F(x) = \text{sgn}(x)\frac{\ln(1+\mu \mid x \mid)}{\ln(1+\mu)}$$

which can be implement as the following Matlab function:

```
function y = ulaw(x)
mu = 255 ;
y = sign(x) .* log(1+mu*abs(x)) ./ log(1+mu) ;
```

You will write the inverse (expansion) function as part of your Pre-Lab report.

The following Matlab function quantizes a signal with a range of ± 1 to (approximately) 8-bit resolution:

```
function y = quant(x)
y = round(128*x)./128 ;
```

To make these functions available to your script you need to save the code above to a file with the same name as the function and a file type suffix of `.m`. For example, save the first section of code above to the file `ulaw.m`. You can also download these two functions from the course web site. The `.m` files should be placed in your working directory.

Pre-Lab

Using the Matlab companding function above as a model, write the corresponding expansion function. The expansion equation is:

$$F^{-1}(y) = \frac{\text{sgn}(y)}{\mu}\left((1+\mu)^{|y|} - 1\right)$$

where $\text{sgn}(x)$ is 1 if x is positive, -1 if x is negative and 0 if x is zero. The Matlab function `sign` implements this function.The Matlab function `abs` implements the absolute value function.

As above, use the element-wise Matlab division, multiplication and exponentiation operators (`./`, `.*` and `.^`) so that your function can handle vector inputs. You may want to test your function against the supplied compression function to make sure they are inverses of each other.

Also answer the following question: If the quantization range is ± 1 and there are 256 levels, what is the expected quantization noise power?

To test companding you will make up a unique nonsense test sentence. The goal is to create a sound segment that is a couple of seconds long that includes both loud and soft sounds. Choose about a half-dozen words from the IPA pronunciation key given above. Alternate consonant and vowel sample words. For example, "buy palm sang boy zoo coin." Include your test sentence in the pre-lab report.

Submit your pre-lab report, *in PDF format*, to the appropriate dropbox on the course web site before the start of your lab.

Procedure

Recording

Connect the microphone (see below) to the PC's microphone input.

Set the recording format: Select `Edit -> Preferences`. Under `Devices -> Recording` set the channels to 1 (mono) and under `Quality -> Sampling` set the sample rate to 8 kHz and 16-bit samples.

Press the record button and hold the microphone close to your mouth while you say the test sentence.

Adjust the microphone gain control while talking so that the recorded signal level is as high as possible without clipping the signal peaks.

Use the cursor to select only the portion of audio you want to analyze. Then remove the rest of the audio as follows: `Edit -> Remove Audio or Labels -> Trim Audio`, then `Tracks -> Align Tracks -> Start to Zero`.

Save the audio samples to a file as follows: `File -> Export` and select `WAV(Microsoft) signed 16-bit PCM`.

Read and Normalize Waveform

Use the Matlab function `wavread()` to read the samples of the recorded waveform:

```
y = wavread('lab2.wav') ;
```

Normalize the samples so that the maximum absolute value is 1:

```
y = y ./ max(abs(y)) ;
```

Plot the result (and save it for your report):

```
plot(y) ;
```

Quantization SNR without Companding

Create a quantized version of the signal and an error (difference) signal:

```
q = quant(y) ;
e = y - q ;
```

Compute the signal power, the quantization noise power (variance) and the quantization SNR without companding:

```
var(q)
var(e)
10*log10(var(q)/var(e))
```

Quantization SNR with Companding

Compress the signal using the mu-law function:

```
u=ulaw(y) ;
```

Plot the compressed signal (and save the plot for your report):

```
plot(u)
```

Does this signal appear different than the uncompressed signal? How?

Repeat the above quantization steps but uncompress the quantized signal before computing the error:

```
q = quant(u) ;
e = y - ulawex(q) ;
```

Compute the power of the quantized compressed signal power, the power of the quantization noise and the quantization SNR as before.

Report

Prepare a report including the following:

- the course and lab number, your name and BCIT ID, the date
- the plots of the uncompressed and compressed signal
- the answer to the question above (how is the compressed signal different?)
- the summary table described below
- a *brief* explanation, with reference to your results, of how companding improved (or didn't improve) the quantization SNR

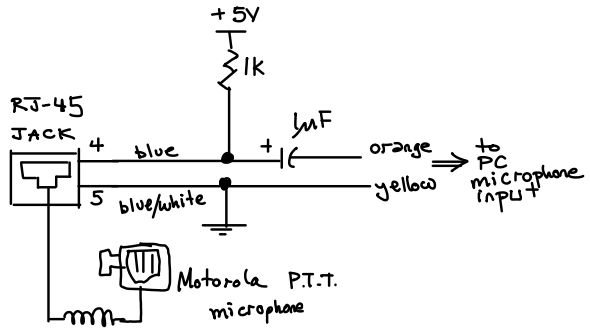
Summarize your results in a table. The table should have four columns: type of measurement, value without companding, value with companding and difference (ratio) in dB. Include the following measurements: signal power, quantization noise power, SNR.

Submit your report *in PDF format* and the `.wav` file containing your digitized waveform to the dropbox on the course web site.

Microphone Interface

The microphones available in the telecommunications lab were designed for use with two-way VHF radios. For historical reasons many of these radios are designed to interface to "carbon" microphones whose resistance varies with the audio waveform. However, PC microphone circuits are designed to work with dynamic microphones that generate small (mV) voltages rather than varying their resistance.

You may have to put together the following simple interface circuit to convert the time-varying microphone resistance to a voltage:



The 1k resistor limits the current and sets the output impedance to approximately the level of the PC input circuit (about 600 ohms). The 1 μF capacitor blocks the DC bias voltage but presents a low impedance to the audio signal. Observe polarity. The components will be available in the lab but you will need a solderless prototyping board, some 24-gauge solid wire and wire strippers.

Pins 4 and 5 of the RJ-45 jack (8 position/conductor) are connected to the microphone's audio circuit. Connect the positive supply to pin 4 (marked solid blue on the connector) and ground to pin 5 (marked blue/white on the connector) as shown below:

Please double-check the voltages and polarity before connecting the microphone. The microphone could be damaged by an incorrect voltage and/or polarity.

If you are recording in mono mode and don't see an audio signal try the other channel on the microphone connector (the violet wire instead of the orange one).