Project Report:
Exploring and understanding basic audio compression with PCM
Multimedia Communications HS 2015

Milan Bombsch
11-917-002

Januar 13, 2016

1 Introduction

1.1 Motivation

PCM is one of the most basic audio encoding techniques. I would like to have a closer look at PCM (and possibly G711) by implementing it myself in Matlab. With this I hope to get a much better understanding of how audio/speech codecs work. Since I have never done anything with audio or signal processing (other than for images) before, I would like to concentrate on the basics and ignore more advanced codecs.

1.2 Approach

I start off by implementing LPCM and doing some test on the performance with different parameters (sampling rate, bit depth). I then try to implement the A-law algorithm for encoding and decoding as described in G711. If I succeed I will again do performance tests and compare it to my LPCM implementation.

1.3 Expected Outcome

The A-law algorithm should perform much better for voice signals, as its quantization rule is specially design for this case. Both LPCM and A-law should lead to audio samples which sound like old telephone calls.

2 Theoretical Background

2.1 PCM

Pulse-code modulation (PCM) [1] is a method to digitalize and store analog signals. This is done by sampling the signal at a specific sampling rate and storing each measured value, which is the amplitude of this signal at this moment. The process of storing an individual sample is done with a specified bit depth. The higher the bit depth the more accurate the signal can be represented. The same holds for the sampling rate. If we know beforehand a feature of the signal we can tweak the sampling rate and bit depth to achieve a good ratio between the bitrate and the quality of the sampled signal.
LPCM (linear PCM) is the most basic form of storing each measured amplitude, since it divides the interval between -1 and 1 into equally sized parts. As we will see in the next section changing this to unequally sized intervals can improve the quality for the same bitrate.

2.2 G.711

G.711 [2] is a standard for PCM for voice signals. It was developed by the ITU-T in 1972. I will not cover the whole standard, but only the basic form without any enhancements. G.711 defines parameters for a nonlinear PCM which is optimized for coding voice signals. It passes signals between 300 and 3400 Hz and uses a sampling rate of 8000 Hz and a bit depth of 8 bit. The standard defines two types of quantization algorithms. The A-Law encoding and the μ-Law encoding. I implemented the A-Law algorithm as it is the one used in Europe.

2.2.1 A-Law Encoding

The A-Law encoding [1] defines a way to encode 13-bit signed integers with 8 bits. This is of course a lossy compression. The compression is done in such a way, that smaller values are represented more accurately, which improves the quality of the compressed speech signal.

3 Implementation

The implementation can be found in folder `code`:

- `uniformQuantize.m` A function which performs a uniform quantization of the input vector with the specified bit depth.
- `aLawQuantize.m` Quantizes input values between -1 and 1 according to the A-Law encoding.
- `changeSamplingRate.m` A function which changes the sampling rate of an input vector. The target sampling rate should be smaller than the current one.

- `lpcm.m` Performs a linear PCM with the specified sampling rate and bit depth.
- `aLawPCM.m` Performs a PCM with A-Law quantization.
- `g711.m` Takes as input an audio vector of floats and performs a bandpass filter (300Hz, 3400 HZ), changes the sampling rate to 8000Hz and performs the A-Law quantization.
- `experiments.m` A script which calculates, displays and saves all plots visible in this report.

4 Experiments

Figure 1 shows a quantized sine wave comparing uniform and A-Law quantization. As an audio sample I used the speech example from the OPUS codec website [3]. This audio file is quantized with 16 bits and sampled at a rate of 48000 Hz. Figures 2 and 3 show this audio example encoded with different bit depths and sampling rates. Figure 2 uses a sampling rate of 48000 Hz. Figure 3 a bit depth of 8 bit. The A-Law Quantization uses in both cases fixed parameters: 8 bits and 8000 Hz sampling rate.

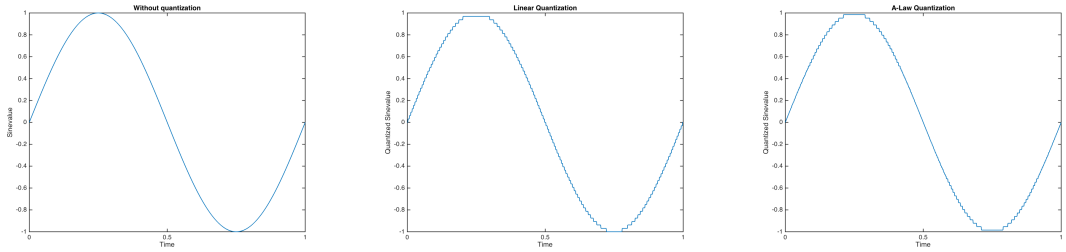

Figure 1: Comparison of a quantized sine function.

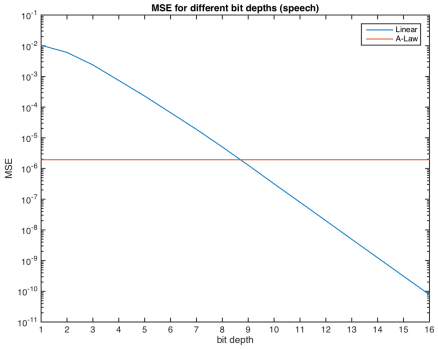

Figure 2: Quantization with different bit depths

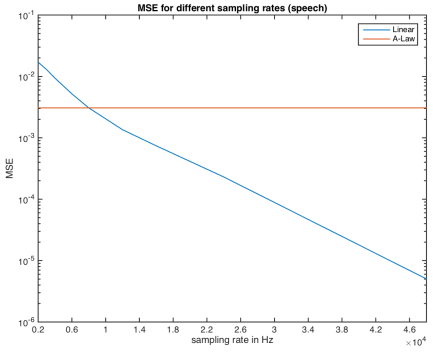

Figure 3: Quantization with different sampling rates

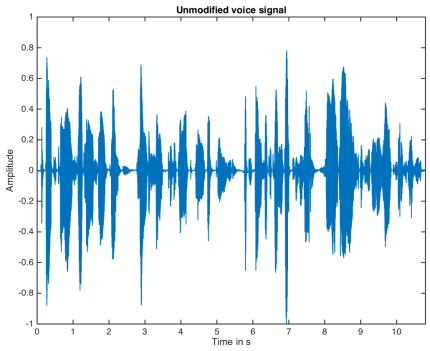

Figure 4: The unmodified speech signal

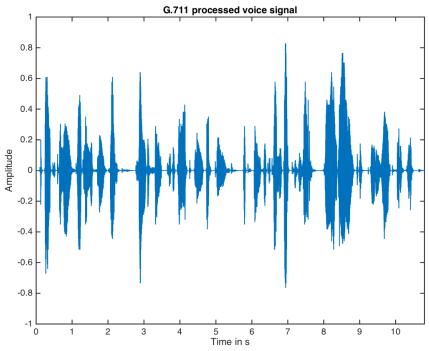

Figure 5: The speech signal process by G.711

5 Analysis

Figure 1 shows a comparison of linear and A-Law Quantization. The linear quantization algorithm fails to produce $+1$ since it uses two's complement representation which quantizes to $[-2^{bitdepth-1}, 2^{bitdepth-1} - 1]$. The A-Law quantization is also designed in such a way that it does not correctly quantize $+1$ and -1. This is fine for speech signals as the peak does not matter but the relative distance between amplitudes over time. We can also see that the linear quantization cases the same error independent of the amplitude, while the A-Law quantization is more fine grained around 0.
Figure 2 shows a comparison of the MSE (Mean Squared Error) at different bit depths for linear quantization. As expected the more bits we use the better we approximate the original signal. As a reference I included the A-Law quantized signal which uses a bitdepth of 8 bits. The 2 lines cross slightly after 8 bits which tells us that the A-Law quantized signal has a lower error. This difference is quite small and does not represent the audible difference between linear and A-Law quantized signals. The A-Law quantized signal is much more pleasant to listen to than the linear quantized one.
Figure 3 shows the same signal sampled at different sampling rates and their MSE. I only used the sampling rates: 2000, 3000, 4000, 6000, 8000, 12000, 16000, 24000 and 48000 Hz to ensure a correct evaluation of the MSE. Otherwise values from the original signal would have to be interpolated. The linear and the A-Law quantized versions cross at around 8000 Hz as this is the sampling rate the A-Law quantization uses. Naturally the more samples we use the lower is the MSE.
Figures 4 and 5 show the speech signal unprocessed and processed by G.711. The processed signal is thinner as the bandpass filter in G.711 removes a lot of frequencies. It also has a lower peak amplitude and overall less variation in the amplitude. Those two changes are audible as a sound which lacks low and high frequencies and has quantization noise. This leads to the sound "effect" of an old telephone.

6 Summary and Conclusion

The A-Law algorithm did not perform much better than linear quantization with respect to the MSE, but the perceived quality and understandability is much better with A-Law quantization. Linear quantization introduces much more audible quantization noise. The G.711 processed audio sounds indeed like an old telephone.

Original
Linear PCM
A-Law PCM
G.711

You need Adobe Acrobat Reader to listen to the audio examples.

References

[1] "Pcm," https://en.wikipedia.org/wiki/Pulse-code_modulation, [Online; accessed 22-December-2015].

[2] "G.711," https://en.wikipedia.org/wiki/G.711, [Online; accessed 22-December-2015].

[3] "Audio examples from the opus website," http://opus-codec.org/examples/, [Online; accessed 22-December-2015].