|  |
| --- |
|  |
| ECSE 490 - Experiment 4 |
| Subband Audio Compression |
|  |
| Group 1 – Isabel Helleur (260350713) and Loren Lugosch (260404057) |
| April 3, 2013 |

|  |
| --- |
|  |

# Multirate Filters

The compression algorithm we are asked to implement in this lab acts on frequency subbands of a signal. We were provided two sets of bandpass filters (each set paired with inverse filters). Each set has a certain number of subbands; in our case, 8 and 32. The plots of several filters in time and frequency for each set are shown below.

## LOT in time

Bandpass filter impulse responses take the form of wavelets (the center frequency damped by exponential functions). These are plots of the first four impulse responses from the filter bank.

## :first_four_LOT_time.jpg

Figure 1 - Four LOT filters in time

## LOT in frequency

Digital filters range from 0 to pi, and since we need a complete set of frequency bands from 0 to the sampling frequency of the input signal, we need uniform bandwidths of pi/(number of bands = 8 or 32). Hence there are 8 evenly spaced bands, which we plot here.

## :first_four_LOT_frequency.jpg

Figure - First four LOT filters in frequency

## :last_four_LOT_frequency.jpg

Figure 3 - Last four LOT filters in frequency

## CMFB in time

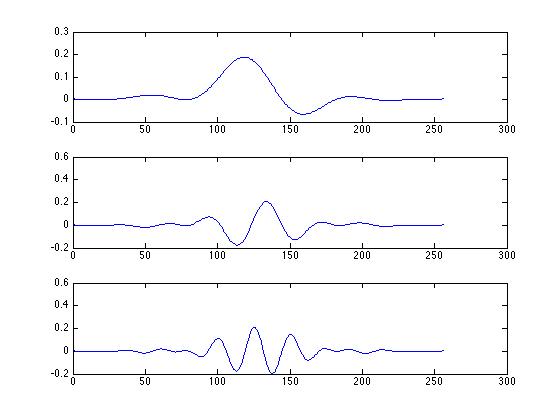
The information here is similar to that of the 8-filter bank, but the CMFB bank has 32 filters. For convenience we plot only a few of them in time and frequency. 

Figure 4 - Three CMFB filters in time

## CMFB in frequency:first_three_CMFB_frequency.jpg

Figure 5 - First three CMFB filters in frequency

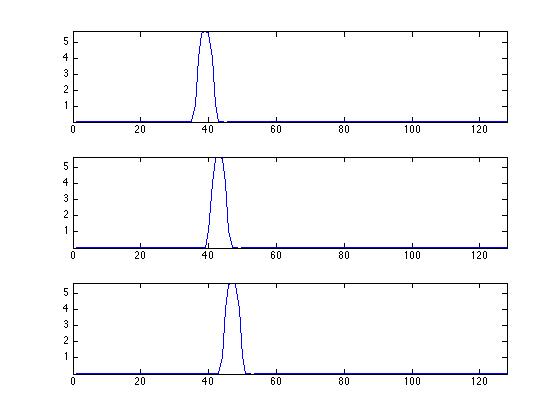


Figure 6 - Middle three CMFB filters in frequency

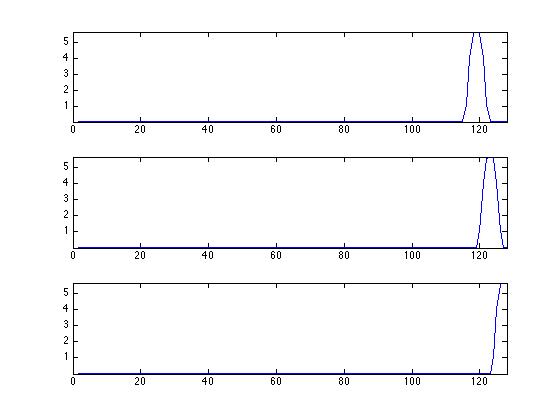


Figure 7 - Last three CMFB filters in frequency

# Analysis/Synthesis

Before we performed any compression, we needed to be able to manipulate different frequency subbands. We wrote a program which uses the provided banks of bandpass filters to separate the signal into those bands, takes the Discrete Fourier Transform of each subband, does (for now) nothing to each of subband, takes the Inverse Discrete Fourier Transform of each subband, and adds each subband in time to reconstruct the signal. To check that the whole procedure works, we compared the output of the program to the original signal graphically, and indeed the program reconstructs the input signal (with a delay/zero padding from convolution with the filters and small scattered noise artifacts from FFT and IFFT) for a variety of waveforms. Since the length of each of the subband filtered signals is equal from the properties of convolution to the length of the input signal plus the length of the impulse response minus one, the delay in the signals was either 256 or 16, depending on the filter bank used.

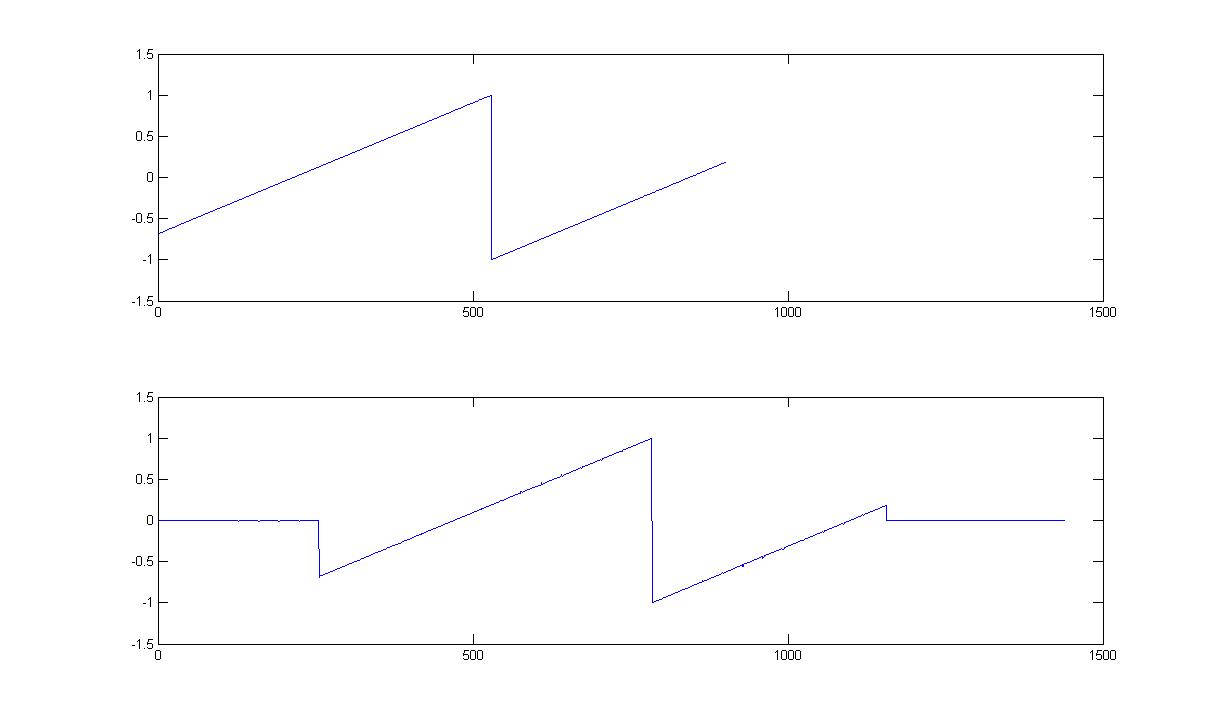


Figure 8 - Sawtooth: Original and Reconstructed

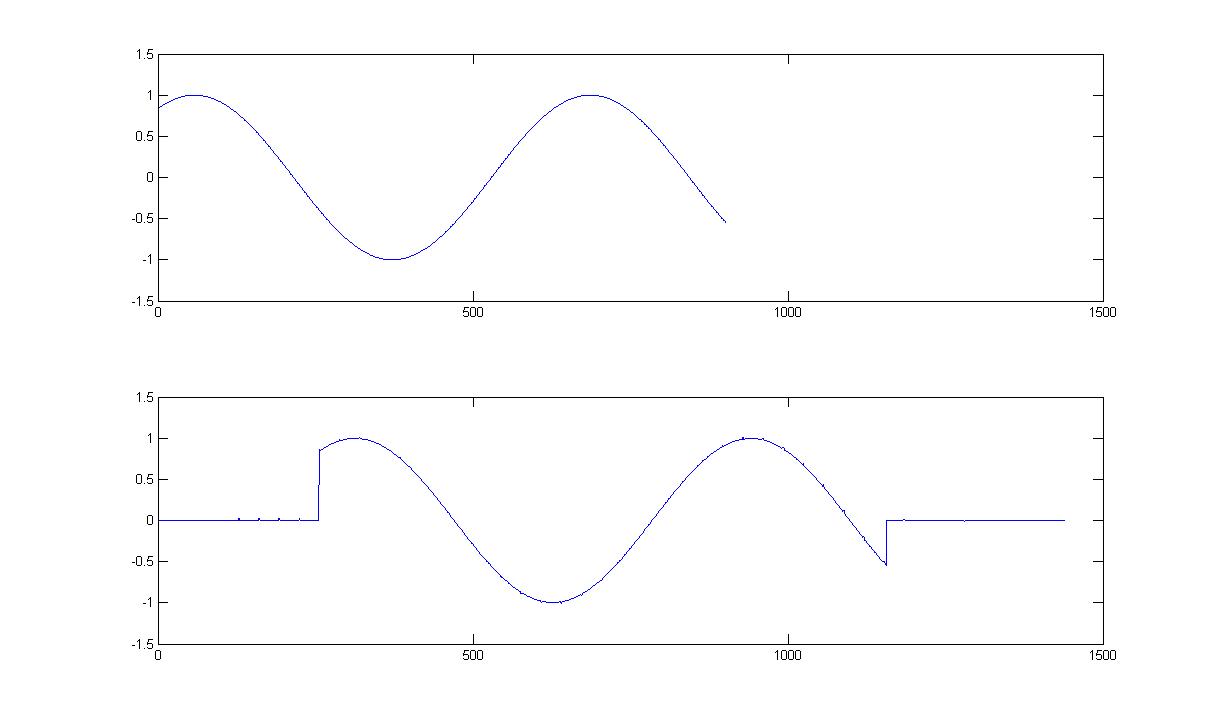


Figure 9 - Sine Wave: Original and Reconstructed

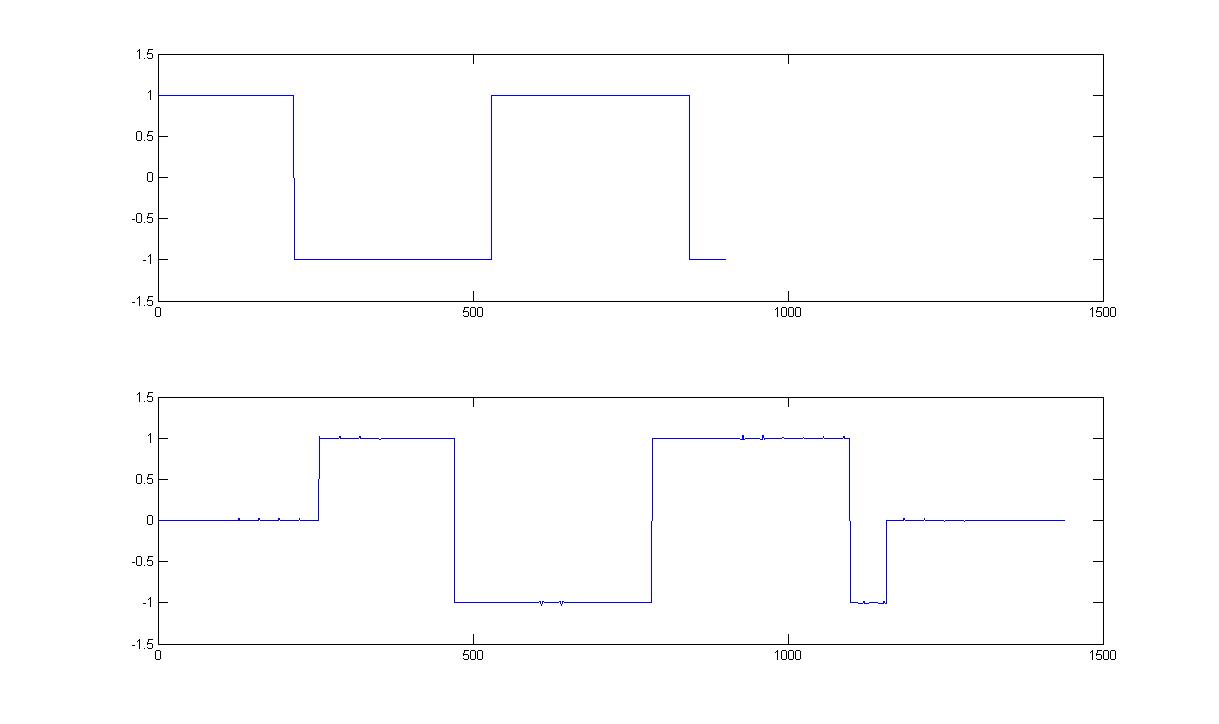


Figure 10 - Square Wave: Original and Reconstructed

# Quantizers

The next component of the compressor is quantization, i.e. forcing the signal to certain values corresponding to a set number of levels, usually in powers of 2 so that the output can be represented in binary words. (Note that the word “quantizing” is easily misused here, as digital signals are already quantized (albeit very finely) and all we’re really changing is the number of quantization levels.) Our quantizer works by scaling the signal to the range -1 to 1 (latching the scaling factor in the process), comparing each value to a set of thresholds at the value of each quantization level, setting the value to the correct level (ceilinged), and rescaling the signal using the inverse of the latched scaling factor.

Implementing uniform quantization (32 levels for each subband), the output of the reconstructor program to a square wave is shown.

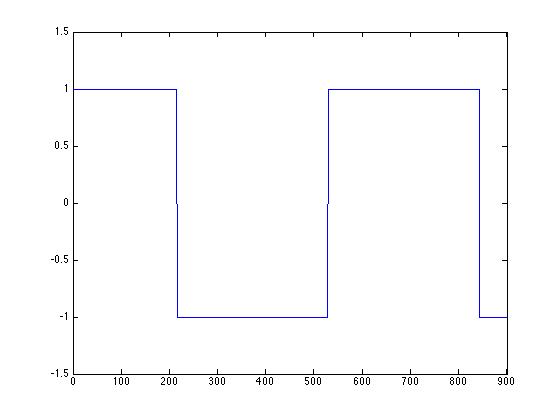


Figure 11 - Input Square Wave

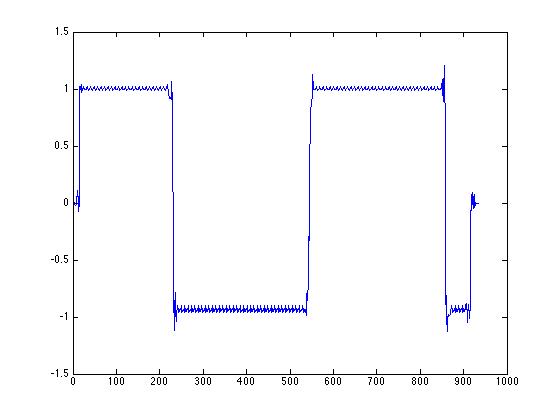


Figure 12 - Uniformly Quantized Square Wave (32 levels each subband)

# Subband compression

## Various rates

“Bit allocation” is choosing the number of bits (and hence the number of quantization levels) to represent a signal. Bit allocation is an important process in data compression because the quality of compression depends on the correct choice of word size.

We decided how many levels to give to each subband by calculating the power of each of the “unquantized” subbands as a rough estimate of the amount of information contained in them. With 8 filters, the values obtained are tabulated below.

Table 1 - Power of subbands starting from 1 going to 8

|  |
| --- |
| 7082.90829152953 |
| 76.6612740351640 |
| 18.6897945569650 |
| 10.6379703120603 |
| 6.08228977322565 |
| 5.21907687003979 |
| 3.87111242685127 |
| 3.93019049616340 |

Since the first subband had the highest power, we increased its quantization levels to 64. The output was no cleaner than that of the schema with 32 levels for the first level (as inspection of the output plot and that of the uniformly quantized signal will show), so we kept the first subband’s levels at 32 with no loss.

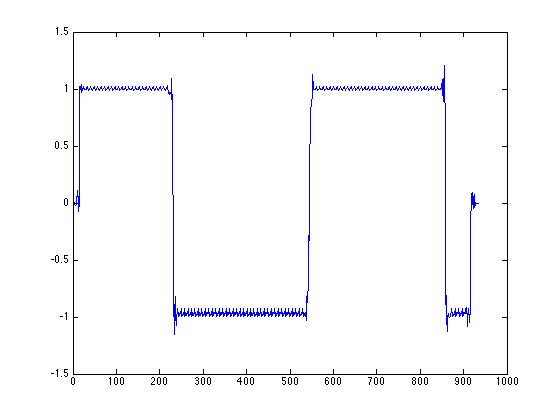


Figure 13 - Quantized Square Wave with 64 levels for subband 1

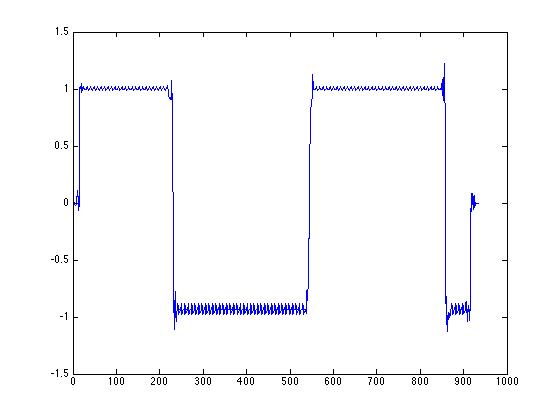
When we decreased the number of quantization levels in the two lowest-power subbands from 32 to 16, there was no noticeable change in the quality of the output signal (even though the SNR was worse); when we went further to 8 levels, the quality diminished greatly in the negative portions of the square wave and in the sound of the music input.

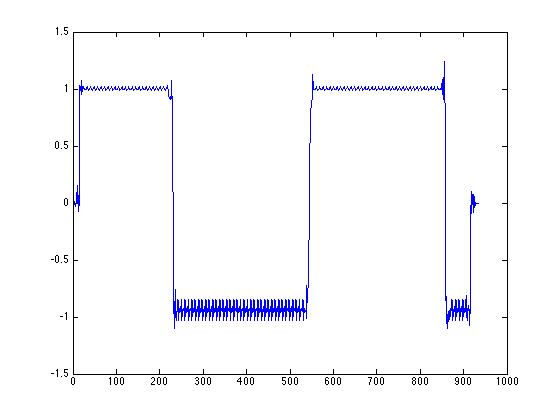
Figure - 16 levels in last two subbands

Figure - 8 levels in last two subbands

## Entropy of each subband signal

To calculate the entropy of the signal, we need the probabilities of the signal taking on each value. We count the number of values at each level and divide that number by the length of the signal to get the experimental probabilities. Since we’re working with doubles, the scaling process incurs a small error, which prevents direct equality checking between the signal value and the quantized levels. We wrote a threshold function which determines whether a value is within “epsilon” of another value (since our errors were always around 0.0001, we made epsilon 0.0002).

We use experimental probabilities in the formula from the lab handout to determine the entropy of each subband. In the 8-filter case with uniform quantization levels (32), the entropies were:

|  |
| --- |
| 3.64028149935995 |
| 3.02350819674586 |
| 3.11936607561678 |
| 3.00515976004611 |
| 3.26228095212576 |
| 3.19645077414915 |
| 2.97603999662340 |
| 3.47066350119182 |

## Direct Quantization of time-domain signal

For comparison with subband quantization, we passed a complete signal (several seconds of classical music) to the quantizer using 16 levels.

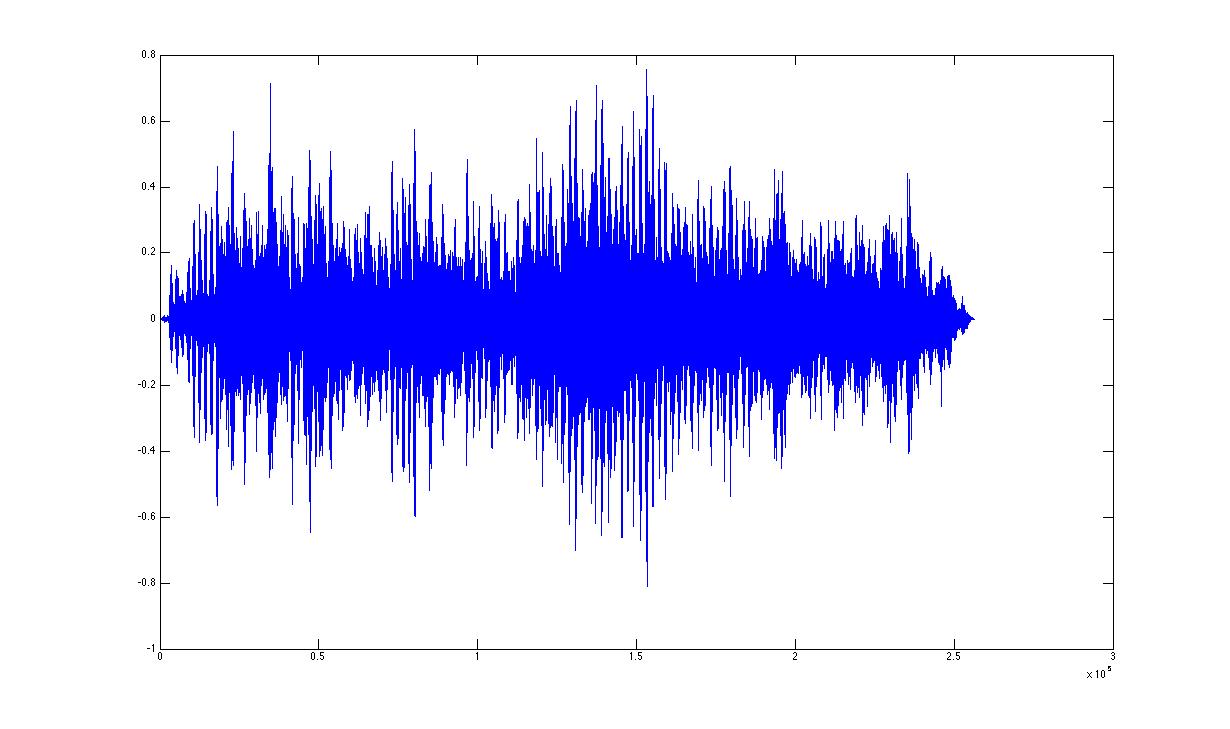


Figure - Non-Quantized Signal

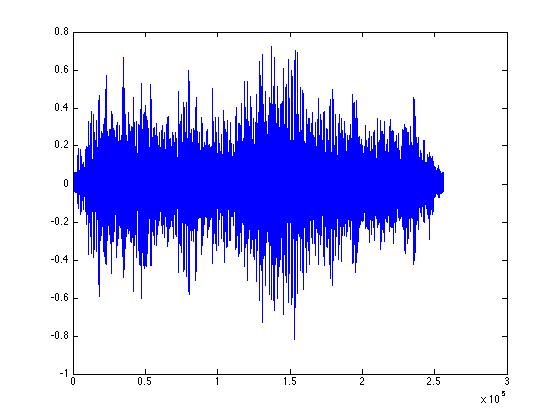


Figure - Subband Quantized Signal

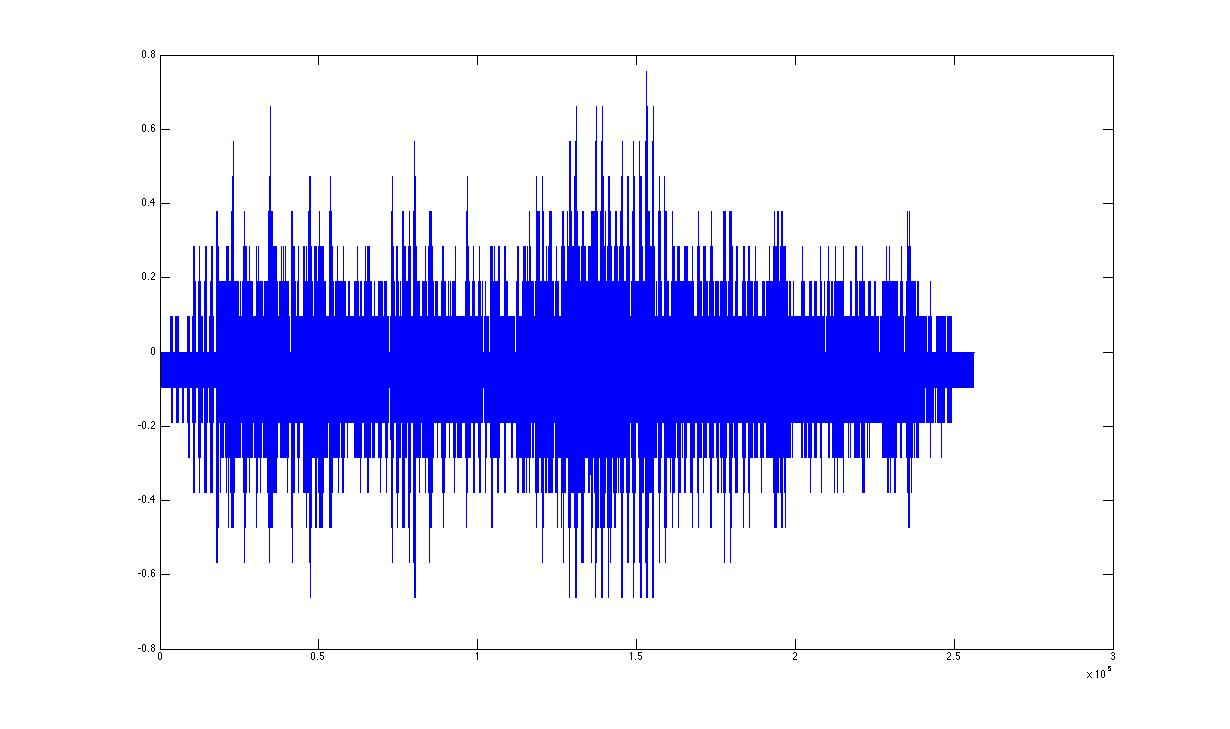


Figure 2 - Directly Quantized Signal

### Signal to Noise Ratio

To measure the quantization noise we subtracted the quantized signal from the original signal. With a few small errors, all values are between 0 and 0.1.

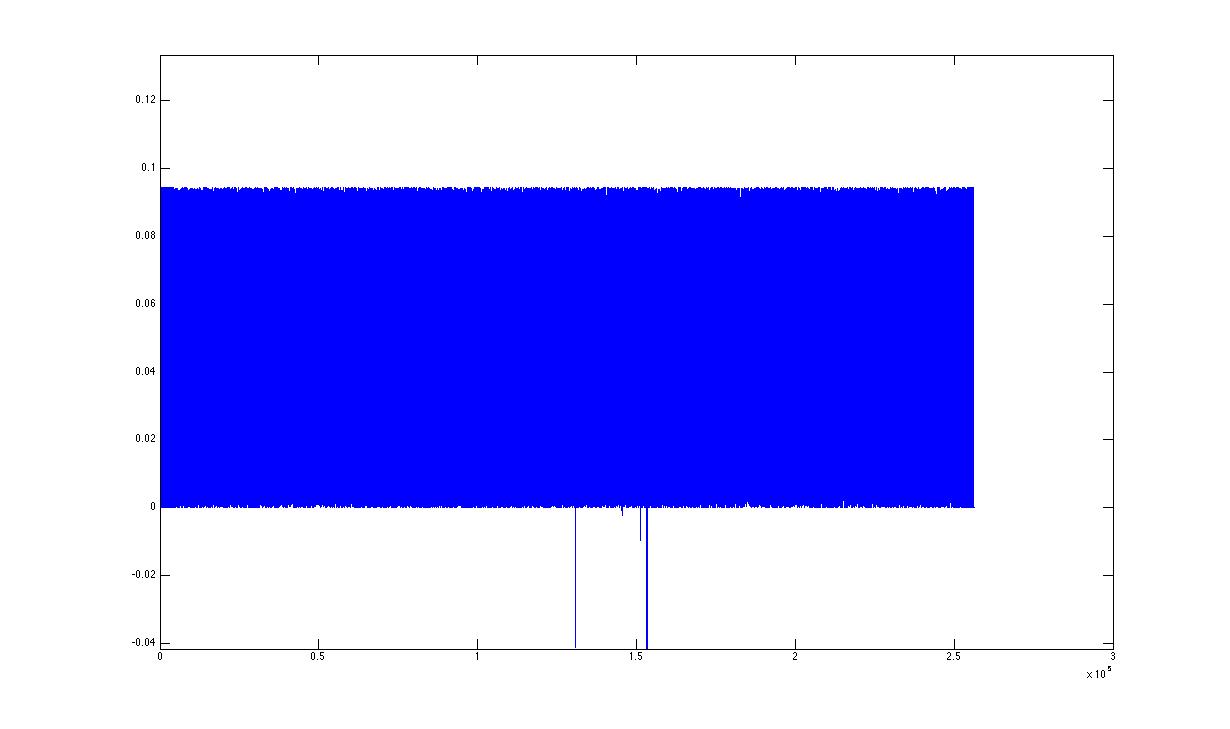


Figure 3 - Quantization Noise (large scale)

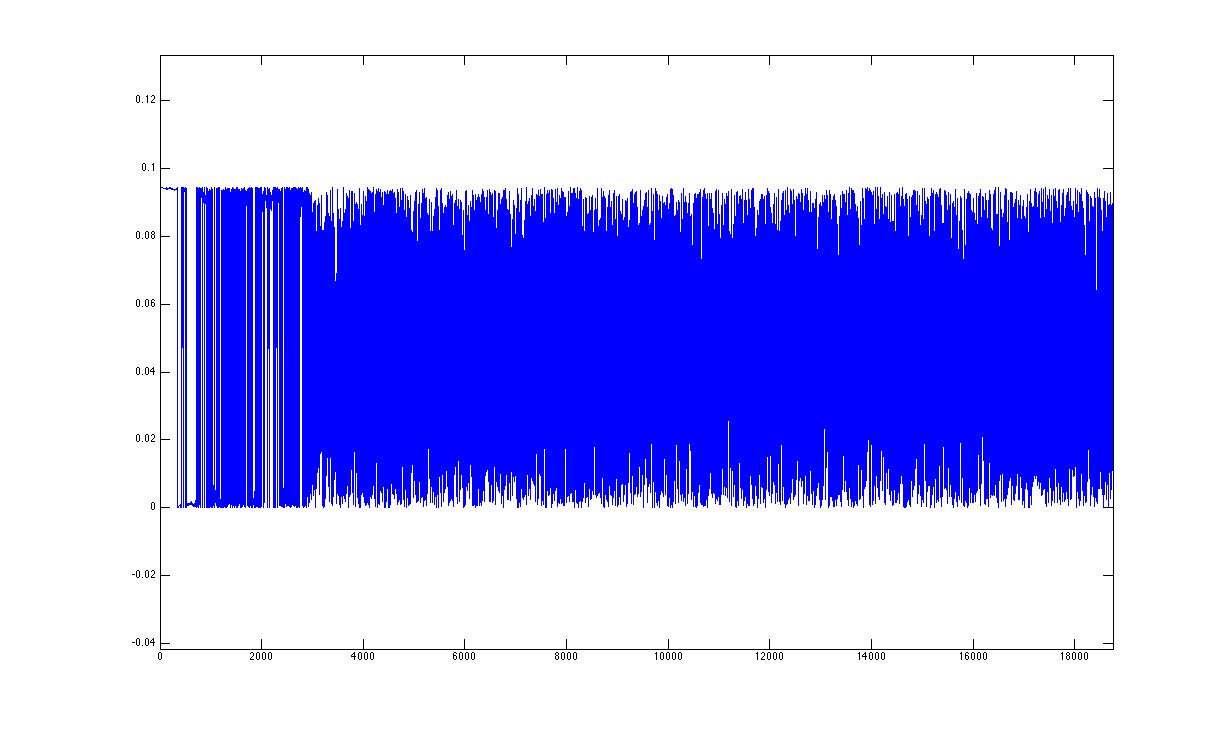


Figure 18 - Quantization Noise (small scale)

## SNR vs. bitrate

The SNR (average power of original signal / average power of quantization noise from subtraction) for 16 levels was 3603.4/ 772.0284 = 4.6674. (This is strongly evident when listening to the quantized signal, which is very noisy at 16 levels). For 32 levels the SNR is 18.7677. This increase in SNR by a factor of around 4 makes sense, since we double the number of levels, and the variance of the noise is inversely related to the square of the level size.