Department of Electrical and Computer Systems Engineering
Monash University

Information and Networks, ECE3141

Lab 1: Quantisation

Authors: Dr. Mike Biggar
Dr. Gayathri Kongara
(updated 9 March 2022)

1. Introduction

Many signals of interest to us originate, and are finally reproduced, in analogue form. Sound and image are obvious familiar examples, since they start as sound pressure levels and light intensities at specific wavelengths respectively and are finally reproduced in a similar form using loudspeakers or earbuds, and TV, computer or smartphone screens. In between these ends, though, we almost always carry these signals as digital signals[1], for the many advantages that digital gives us and which we will explore in lectures.

Conversion of these analogue signals to digital form requires sampling the signals periodically in time (and, in the case of images and video, in space as well), and representing each sample value as a number with finite accuracy. Here, we focus on the accuracy with which values at each of the sample points are represented, and how we can represent the set of all the sample values most efficiently.

Quantisation does not always involve conversion from an analogue input. It can involve any sort of operation to represent a quantity more coarsely, as one of a discrete set of possible values. In particular, a signal may already be in digital form but at very high resolution and we may decide to represent it less accurately (e.g., to save bits in its representation). This, in fact, is what we will be doing in this laboratory; we will start with signals already digitised at high definition (large number of bits per sample) and quantise them to a lower definition (fewer bits per sample).

In this laboratory class, you will carry out the following as you investigate the effect of quantisation and sample representations of one-dimensional signals:

- Understand some of the decisions involved in designing a quantiser.
- Listen to the effect as you vary the number of quantisation steps used to represent a sinusoidal signal, as well as segments of recorded sound (speech and music).
- Measure the SNR resulting from quantisation of audio signals as you increase the accuracy of the representation, and compare the result with theory
- Develop your skills in Matlab and audio processing

Note on Matlab scripting

You should make sure you understand the Matlab processing you are doing and are encouraged to write your own code to carry out the task required. However, you have limited time and do not want to spend it all debugging, so example script segments are provided in

[1] The most familiar exceptions are AM and FM radio, which are still analogue. All TV broadcasts have now moved to digital, and with the emergence of DAB and DAB+, radio is moving in the same direction.

this lab document, so you can use these to help if you want to. You can either copy and use them, or just get hints about how to use certain functions – whatever works for you. In any case, you still need to stitch the pieces together and add extra code to make it all work, so you need to make every effort to understand the processes being carried out.

2. Pre-lab

Prior to the laboratory class, you must read through this entire laboratory description and complete the pre-lab online quiz. You may also be able to complete answers to some of the questions below which are based on theory (i.e. that don't depend on experimental results).

You should also consider each exercise given, and think about the Matlab processing you need to do (that is, how will the algorithm work? What instructions will you need? Do you know how to use them?).

If you want to use your own audio clips (e.g. a short segment of music), you need to organise this (that is, extract a short segment from the original audio) in advance of the lab class. However, you MUST respect copyright as described in the next section.

3. Copyright

The audio clips you will be using are short segments of copyrighted content. You may use them for your laboratory investigation, but must then delete all copies, whether original or processed, and must not under any circumstances publish (in any sense, including posting on social media) any version (even if it is highly distorted from the original) of these clips.

If you are bringing and using your own audio clips, these MUST have been obtained legally according to copyright laws. The above restrictions (deleting all copies when you've finished, not publishing or distributing any copies) also, of course, apply.

4. Background theory: quantifying quantisation

Figure 1 shows a continuously varying 1-D analogue signal that is sampled periodically, and where the value at each sample point is quantised. (Of course, the quantisation is very coarse in this figure to allow the differences to be seen; ordinarily the quantisation noise would be much smaller.)

Suppose the original analogue function is $y=f(t)$ defined for all t. (This is the red curve in Figure 1.)

$y_s=f(nT)$ is the sampled sequence, where T is the sampling interval and n is an integer. y_s is defined only for $n \in I$ (an integer). So, for example, $y_s(1.7T)$ is not defined; it has no representation in the sampled domain. Though y_s is a sampled signal, it is not yet quantised; y_s can still have any value within some bounds. For instance, $-V \leq y_s \leq V,\ y_s \in R$.

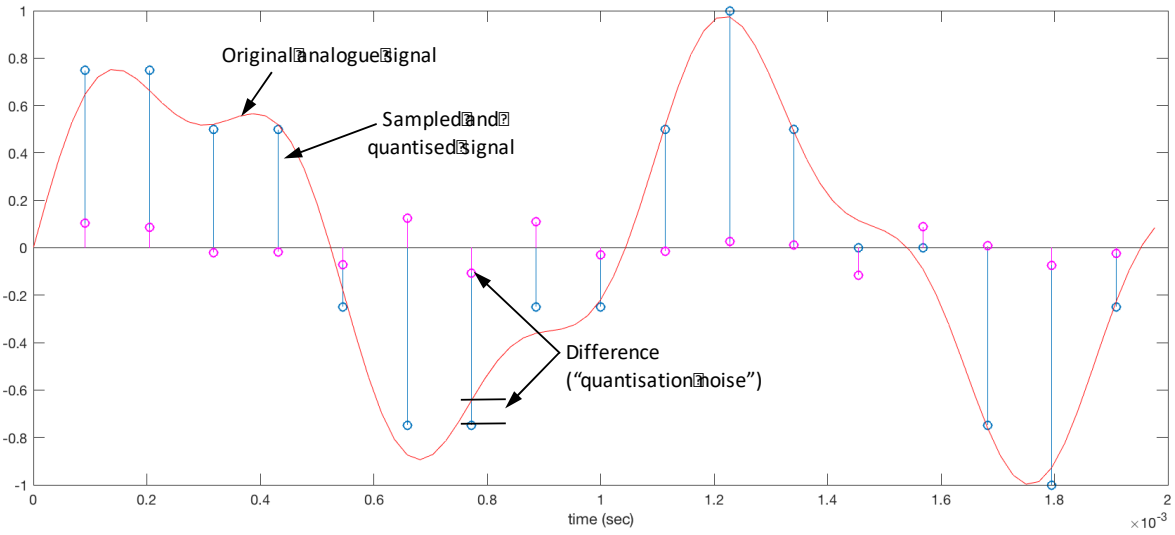


Figure 1. Sampling and quantisation of a one-dimensional analogue signal.

If we are to represent the y_s in a digital system, then we must use a finite number of bits to represent each sample value. That means that we must "round" the values of y_s to a subset of allowable output values using some form of quantisation. A linear quantiser[2] is shown in Figure 2. As a transfer function which maps the input (horizontal axis) to the output (vertical axis), whatever value the input might have, the output will be restricted to one of a set of discrete (quantised) values.

To make sure you understand this quantisation function, consider how the output would change as the input varies continuously, taking on every value on the horizontal axis.

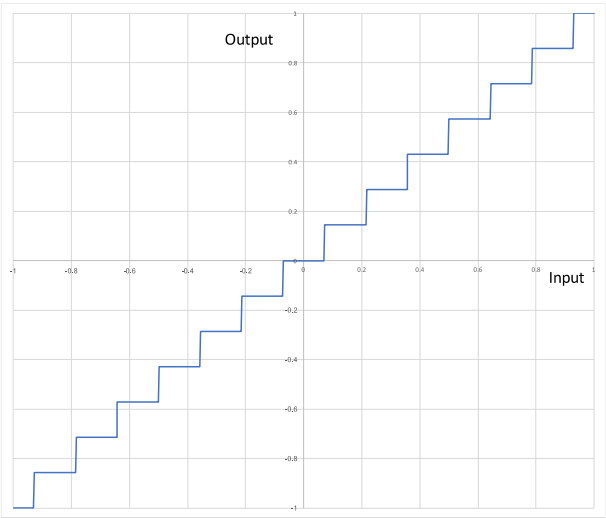


Figure 2. An example of a linear (or uniform) quantiser transfer function. This is a "uniform, midtread" quantiser.

An obvious, and important, consideration is the difference between our original signal and the quantised, approximate representation. To characterise our quantised sequence (let's call it y_q), we could think of it as a distorted version, or an approximation, of the unquantised values, i.e. $y_q = \hat{y_s}$. But it is often more useful to consider it, or model it, as the sum of the

[2] Quantisers need not always be linear. An example is if we wanted the introduced error (or quantisation noise, as it is defined in the coming paragraphs) to be smaller for smaller signal levels (so the ratio of error to signal is about the same for all signal magnitudes). In this case, the quantiser steps would be smaller closer to the origin. In fact, this is exactly what is done in telephony.

original, exact *y(nT)* values, to which some noise (specifically, "quantisation noise", n_q) has been added.

$$y_q(n) = y_s(n) + n_q(n) = y(nT) + n_q(n)$$

Why is such a representation useful? Because, if we can characterise this signal distortion or noise we're calling n_q, then we can calculate its power and compare it with the original signal, to obtain a SNR (Signal to Noise Ratio)[3]. Then we can select the parameters that influence the size of the quantisation distortion, so that the SNR is kept within defined limits.

4.1. Forms of linear quantiser – mid-tread and mid-rise

Note that one of the decisions to make when designing a quantiser is what to do at the origin. In Figure 2, the origin (0,0) is in the middle of a horizontal "tread". The (obvious) alternative is to put the origin in the middle of a vertical "rise"[4], as shown in Figure 3.

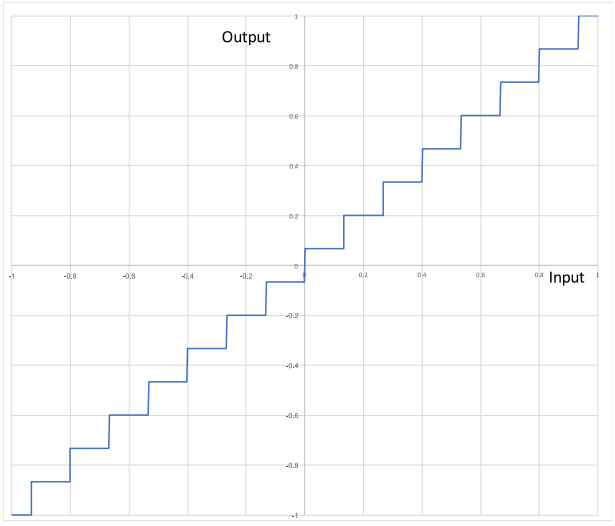


Figure 3. A variation on the linear (or uniform) quantiser from Figure 2. This is a "uniform, midrise" quantiser.

What are the advantages and disadvantages of the "midtread" and "midrise" quantisers? (Hint: you can answer this simply by thinking about the plots in Figures 2 & 3. You do not need to go searching for some extra theory.)

Both "midtread" and "midrise" quantizers are two types of uniform quantizers (the representation levels are equally spaced). The "midtread" quantizers levels are even in number. The "midrise" quantizers levels are odd in number.

Midtread
Advantages:

- **The Midtread quantizer has a zero level. This means that it can represent a zero-amplitude signal with a higher accuracy.**
- **There would also be less noise as the noises with low amplitudes can be filtered off easily with a midtread quantizers.**

[3] To avoid confusion with random (e.g. thermal) noise, sometimes we use the term SQNR ("Signal to Quantisation Noise Ratio") instead.
[4] In the building industry, the terms "tread" and "riser" are used to refer to the horizontal and vertical components of stairs. The analogy is obvious!

Disadvantages:

- **It consists of an odd number of levels. This will result in the inefficient and underutilization usage of bits and quantization levels, leading to more quantization noise.**
- **There could be a large magnitude of quantization error in the lower level.**

Midrise
Advantages:

- **The mid-rise quantizers will have even number of levels. This will increase the efficiency of the quantizer as it is able to utilize most if not all of the possible levels, leading to less quantization noise.**

Disadvantages:

- **The midrise quantizers does not have a zero level – meaning that a weak signal will need to use the first level of quantizers to map to (since there is no zero level). The signal which have an amplitude of zero will be represented inaccurately.**

The difference between the mid-tread and mid-rise quantisers is noted for interest, but we will concentrate in this lab on the use of the more common mid-tread quantiser. (If you have the time and interest, you are welcome to try both, though.)

4.2. Modelling quantisation noise[5]

Here we obtain a quantitative measure of the SNR due to quantisation, making some simplifying assumptions.

Suppose that our signal before quantisation is bounded to the range $-V \leq y_s \leq V,\ y_s \in R$, and that quantisation of that signal involves setting the output y_q to one of M allowable levels. If our quantiser is uniform, then the step size between the allowable levels will be a fixed value

$$\Delta = \frac{2V}{M} \qquad (1)$$

We further assume that M is a power of 2, because that means that we can represent each output level with an m-bit word ($M = 2^m$), without wasting bits[6].

Now, if the step size between output levels is Δ, then a moment's thought (after looking at the staircase function of Figures 2 or 3) will tell us that the largest possible error (provided the input does not exceed the limits of $\pm V$) between input analogue signal and output quantised signal will be $\frac{\Delta}{2}$. In fact, the difference between input and output values will have the form shown in Figure 4.

[5] This mathematical development of the relationship between noise due to quantisation and the number of bits *m* used to represent the quantisation levels is the same as that presented in the week 1 theory module "Quantisation".
[6] If M was not a power of 2, then some of the m-bit words would never be used. We could have, for instance, M=21, which would need m=5 bits to distinguish which of the 21 possible output levels we obtain, but there would be 11 possible 5-bit words that would never be used ($2^5 = 32$).

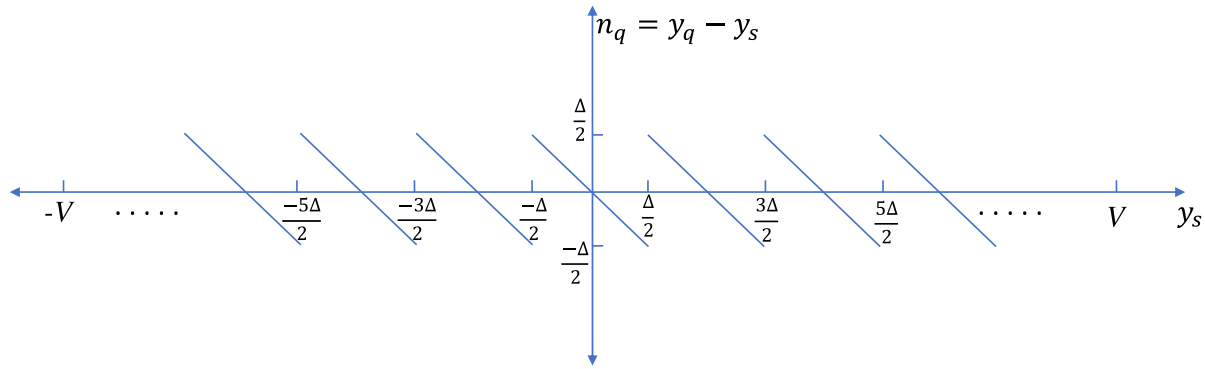


Figure 4. Error signal as a function of input.

Clearly if *m* (and therefore *M*) increases, then Δ will become smaller and so the amplitude of the error signal will also become smaller. We would like to know not just the amplitude of the error, but its power, because that would allow us to obtain the power ratio of SNR.

We can estimate the noise power by using a statistical argument as follows. If we are quantising an input signal that is non-deterministic (e.g. an audio signal that could be music or singing), then we cannot calculate in advance what the noise signal will be, but we can deduce something about its statistical nature. We know that the probability density function of n_q will be zero outside the range $\frac{-\Delta}{2} \leq n_q \leq \frac{\Delta}{2}$, but what about inside that range? A particular input value y_s could be anywhere on the horizontal axis of Figure 4. However, if we assume that *M* is relatively large, there is no reason to assume that any value of n_q within the possible output range is any more likely than another[7]. Therefore, we will model the pdf (probability density function) as constant within this range and exactly zero outside it, as shown in Figure 5.

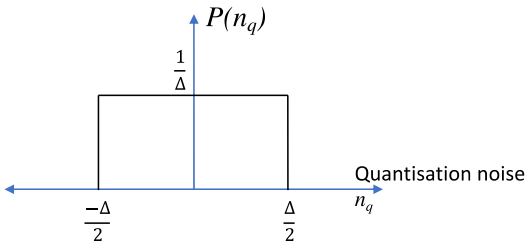


Figure 5. pdf of quantisation error signal.

To calculate the average noise power, we need the mean square error, or the expected value of the square of n_q. That is (substituting $n = n_q$ to keep the maths uncluttered)

$$Quant.noise\ power = \int_{-\infty}^{\infty} P(n)\,.\,n^2 dn = \frac{1}{\Delta}\int_{\frac{-\Delta}{2}}^{\frac{\Delta}{2}} n^2 dn = \frac{1}{\Delta}\left[\frac{n^3}{3}\right]_{-\frac{\Delta}{2}}^{\frac{\Delta}{2}} = \frac{\Delta^2}{12} \qquad (2)$$

We have an expression for Δ from eq. (1) that we can substitute into eq. (2), so the noise power is

$$Quant.noise\ power = \frac{\Delta^2}{12} = \frac{V^2}{3.M^2} = \frac{V^2}{3(2^{2m})} \qquad (3)$$

[7] There are some particular circumstances when this would not be true. Can you think of an example? In any practically useful situation, though, this is a reasonable assumption.

The final part of the derivation is to introduce the signal power. We do not have a value for it, but we know that the power is the square of the standard deviation σ_y. Therefore, substituting the quantisation noise power from eq. (3), the SNR is

$$SNR = \frac{Signal\ power}{Noise\ power} = \frac{3.{\sigma_y}^2(2^{2m})}{V^2} = \left(\frac{\sigma_y}{V}\right)^2 3(2^{2m})$$

And, if we calculate the SNR in dB, we have

$$SNR_{dB} = 10log_{10}\left\{\left(\frac{\sigma_y}{V}\right)^2 3(2^{2m})\right\} = 6.02m + 10log_{10}\left\{3.\left(\frac{\sigma_y}{V}\right)^2\right\}$$

We can obtain an expression with the single variable *m* if we can get a relationship between the signal standard deviation σ_y and the maximum amplitude *V*. This relationship is obviously signal dependent, but we can obtain a result through measurement of "typical signals". In particular, we find that speech signals typically have a ratio of peak to standard deviation of about 4:1. In this case,

$$SNR_{dB} = 6.02m + 10log_{10}\left\{3.\left(\frac{1}{4}\right)^2\right\} = 6.02m - 7.27\ dB \qquad (4)$$

So, after all this derivation, we see that the SNR due to uniform (or linear) quantisation has the following features:

- The SNR in dB is linearly related to *m*, the number of bits used to represent each sample value
- The intercept of the linear relationship depends on the ratio of peak to standard deviation of the signal being quantised
- The slope is independent of the signal and shows that we increase the SNR by about 6 dB for every extra bit of accuracy of representation of sample values. For instance, if we change from using 8-bit words to represent sample values, to using 10-bit words, then our SNR will improve by about 12 dB.

5. Quantisation distortion with a sinusoidal test signal

a) Let us start our investigation of quantisation using a simple sinusoidal test signal. In Matlab, create a vector that represents a time-varying sinusoid of, say, 1 kHz, that lasts for a couple of seconds[8]. Choose a suitable sampling frequency to use to represent it[9]. If you generate the corresponding time vector (that is, a vector that contains the times at which each of the sinusoid values is calculated), you will easily be able to plot one against the other later if you're interested to do so. Play the vector (using the *sound()* command) so you can hear (and check) it.
Here is some example code in case it helps:

```
TestFrequency_Hz = XXXX;
SampleFrequency_Hz = YYYYY;
Sampling_Period=1/SampleFrequency_Hz;
Amplitude = 1;
Duration_sec = 2;
```

[8] You could carry out this and the following steps without creating an m-file, but to save you retyping and to allow for mistakes, it would be best to use an m-file.
[9] You might be tempted to sample a 1 kHz sine wave at the Nyquist rate of 2 x 1 = 2 kHz. However, if you give a little thought to which parts of the waveform are sampled if there are exactly 2 samples per cycle (the first at t=0), you'll see why we need to sample at a rate GREATER than the Nyquist rate.

```
Num_Samples=SampleFrequency_Hz*Duration_sec;
time_vec=(0:Num_Samples-1)*Sampling_Period;      % Time vector
signal_vec = Amplitude*sin(2*pi*TestFrequency_Hz*time_vec);      %
Signal vector
sound(signal_vec,SampleFrequency_Hz);
```

With reference to the last line above, note that if the sample rate is not specified (i.e. if you just use *sound(signal_vec);*), the data will be played out at a default sample frequency of 8192 Hz. Then 1000 Hz won't sound like 1000 Hz! (Try it!)

b) Now let's quantise this sinusoid to varying degrees of accuracy. Add some code to your Matlab script to quantise (with a uniform, mid-tread quantiser) the sinusoid so it is represented by a restricted number of levels[10]. The number of levels should be determined by a parameter *m*, which is the number of bits needed to define which level it is. That is, with *m* bits, we can represent up to 2^m quantisation levels. (Hint: Since we're talking about rounding off numbers, so each is represented by a number from a smaller set, then the *Matlab* functions *round*, *ceil* and *floor* would seem to be good candidates for use.) You should develop some way to check that the quantiser is doing what you expect it to do before proceeding.
Again, some example code is provided below. This expects an input in the range (-1,1) and the quantised output is in the same range.

```
% midtread quantiser
>> Num_levels=2^m-1; % Note: odd number of levels
>> Quant_vec = 2*round(signal_vec*(Num_levels-1)/2) / (Num_levels-1);
```

c) When we quantise, we want to measure the magnitude of the error introduced, so create a quantisation noise vector, which is the difference between the original signal vector and the quantised version. Calculate the SNR due to quantisation, in dB[11].

How will you calculate SNR in dB? (That is, what is the formula and what Matlab code will do it?)

Firstly, we must obtain the power of the signal with the following formula:

$$P_s = \frac{1}{N} \sum_{i=1}^{N} S_i^2$$

We can obtain the noise from the following formula:

$$P_s = \frac{1}{N} \sum_{i=1}^{N} (S_i - \widehat{S_i})^2$$

where $\widehat{S_i}$ stands for the distorted signal and S_i stands for the original signal

[10] Matlab provides specialised calls to do quantisation, but there is value in doing it "from first principles", to make sure you understand exactly what is going on.
[11] In the experimental parts 5 & 6, you are NOT plotting the line 6.02m-7.27 dB. If you plot a line, it doesn't make much sense to then consider "is it linear?'. Instead, you have generated a signal vector, quantised it and can calculate the relevant signal and noise power. You are asked to find (or measure) the actual SNR. THEN you will consider how well it matches the theoretical model that leads to the 6.02m-7.27 formula, what the underlying assumptions of that model are, and whether they are valid.

We can then proceed to use the SNR formula given in the lecture notes.

The formula is SNR =10log$_{10}$(P_S / P_N), where S is the mean square of the signal and M is the mean square of the Noise.

The MATLAB code is

```
Quant_vec = 2*round(signal_vec*(Num_levels-1)/2) / (Num_levels-1);
Quant_noise = Quant_vec - signal_vec;
Sig_Pow = sum(signal_vec.^2)/Num_Samples;
Qnoise_Pow = sum(Quant_noise.^2)/Num_Samples;
Pow_ratio = Sig_Pow / Qnoise_Pow;
SNR = 10*log10(Pow_ratio);
```

Assuming that the Sample Frequency used is 2.3kHz, the value of the SNR obtained would be 8.9088dB for two bits.

d) Now, as you increase m from 2 up to 16, listen to the quantised sinusoid and record the SNR you obtain.

At what SNR value (and with how many bits m) can you no longer tell the difference between the quantised and original signal when listening to it? Does the answer change depending on whether you're using headphones or speakers? Why?

Bits	SNR in dB
2	8.9088
3	17.9228
4	26.1153
5	33.7175
6	36.8944
7	44.5720
8	50.6011
9	54.6111
10	59.3565
11	66.8567
12	73.5976
13	78.3515
14	87.8925
15	93.8948
16	97.6150

The listener would not be able to hear the difference at around 8 bits for headphones and 4 bits for speaker

Quantization of noise will be more evident if headphones is used since headphones suppresses unwanted external noise from the surrounding.

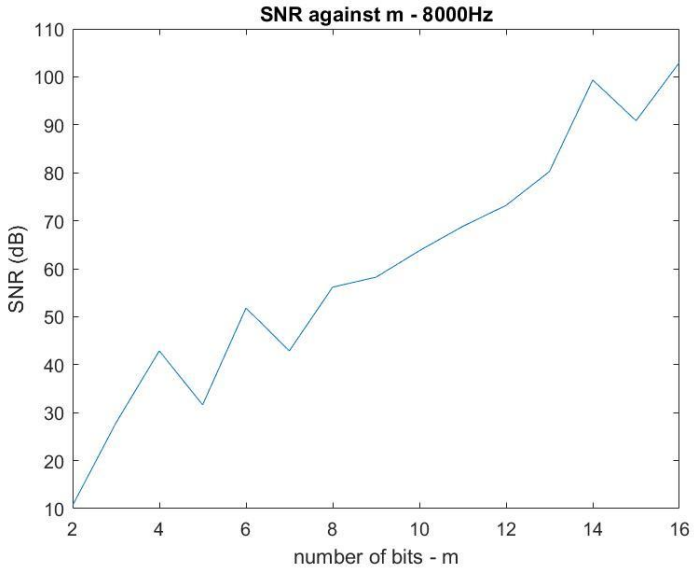
e) We want to explore the shape of the function that describes quantisation noise SNR as a function of m.

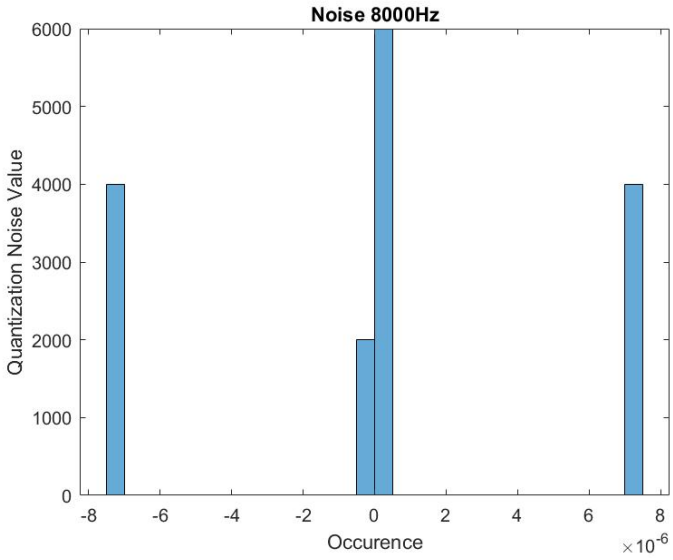
What does theory (from lectures, and Section 4.2 above) predict will be the relationship between SNR and m?

Based on the theory, the relationship between SNR and m will be 6.02m – 7.27dB, where m is the number of bits. Based on the equation above, we can deduce that SNR will increase linearly with m.

f) Choose a sampling rate that is a small integer multiple of the signal frequency (e.g. 8 kHz) and plot SNR against m, for m = 2 to 16.

Copy your plot here. Does it have the form you'd expect (that you described in part e, above)? Why? (Hint: If you're having trouble figuring out what is going on, plot the histogram of the quantisation noise vector. This is very straightforward; just `histogram(Quant_noise);`)

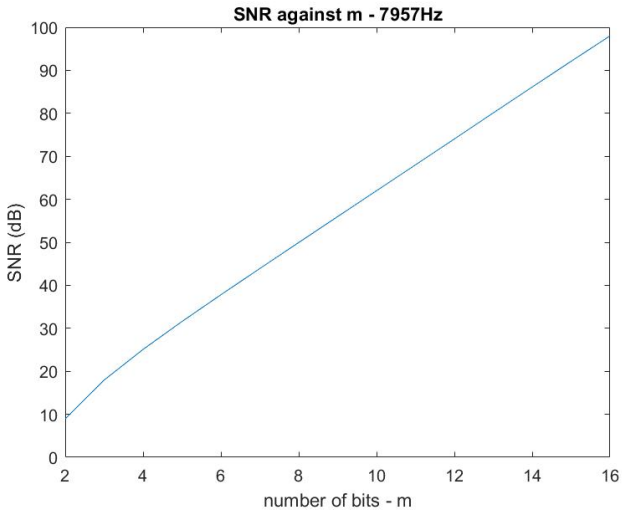


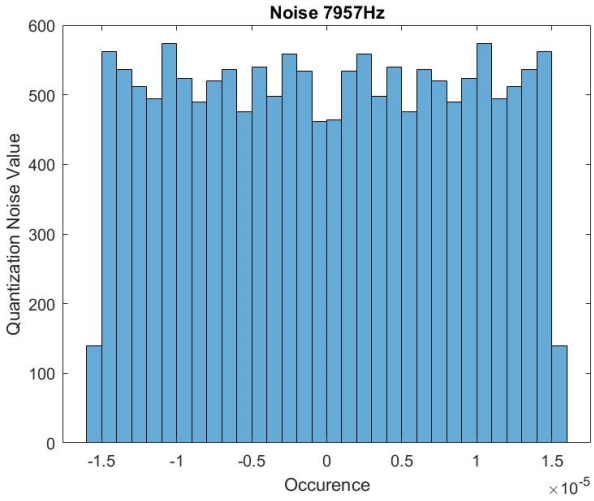


The plot shown does not match the form I have expected. The assumption made that the quantization noise is uniformly distributed is incorrect. As we have seen in the histogram above, the quantization noise is not uniformly distributed. We can also see that there are noise values with high frequencies in the histogram. This could explain why we could not obtain the theoretical form.

g) Change the sampling rate so it is near to that above but NOT an integer multiple of the test frequency (e.g. 7957 Hz). Repeat your plot of SNR vs *m* for this case.

Consider again the plot of SNR vs *m*. What do you find, and can you explain it?

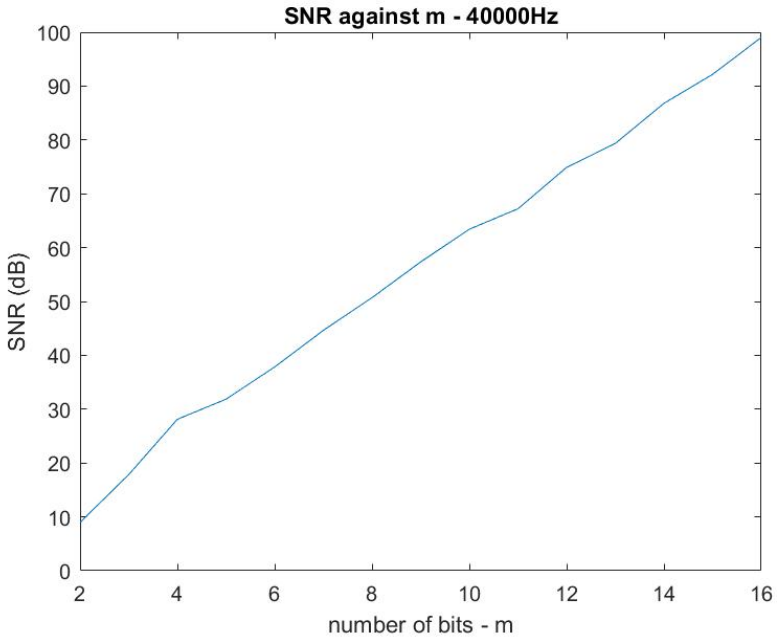


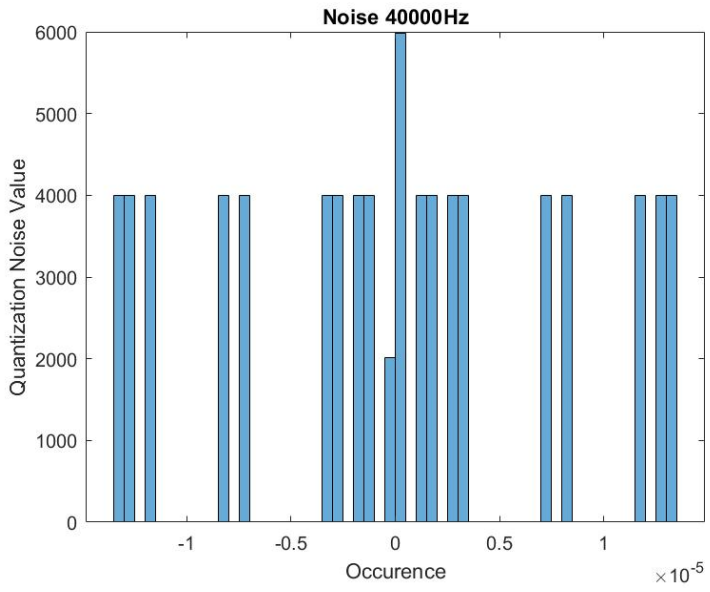


The form that we have obtained is more like a linear relationship compared with the form we have obtained from the question above. The sampling rate has been changed so that it is not an integer multiple of the test frequency. This would result in the quantization noise having a uniform distribution. The noise value will be different and therefore the graph will look much more linear compared to the plot before.

h) Now change the sampling rate so it is a large integer multiple of the signal frequency (e.g. 40 kHz). Repeat your plot of SNR vs *m* for this case.

Consider again the plot of SNR vs *m*. What do you find, and can you explain it?





The form that we have obtained isn't a perfect fit. There is some form of uniform distribution between the quantization noise values. Since the sampling period of the signal will be small, it will produce more variation in the quantization noise as compared to the small integer smapling frequency.

i) What have you learned about the relationship between the signal frequency and sampling rate if our model (from section 4.2) is to be reliable?

The sampling rate must not be a small integer multiple of the signal frequency if our model is to be reliable.

j) Before we get on to investigate it, is the observation you just made also going to occur when we quantise real world signals (e.g. audio signals rather than just sine waves)? Why?

No, the observation I have just made is not going to occur when we quantise real world signals. This is because real world signals will not be perfect as a perfect sinusoid due to the presence of external noises.

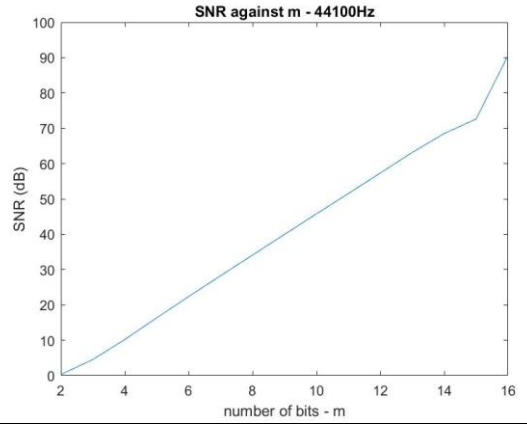
6. Quantisation distortion with real audio signals

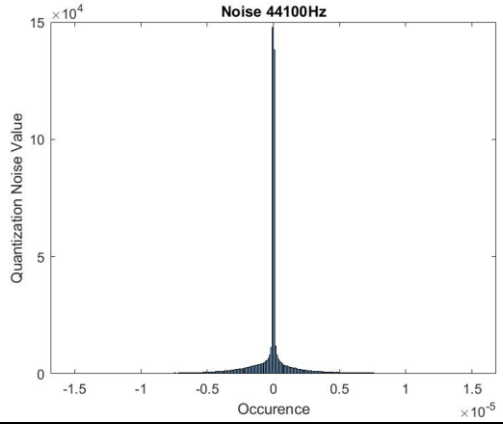
a) Now that you've succeeded with a test signal, let's repeat the exercise with an audio sample. Modify your script to read in an audio file (use both speech and music). The audio files *ShortClip_Shine On.wav* and *12secSpeechSample - Speeches That Changed The World - Intro.wav* are provided for you on Moodle or, as noted earlier, you may have brought your own clips. (Note the copyright information at the start of this laboratory description, and make sure you respect the request to delete all copies when you are finished.) The provided files are digitised to 16-bit accuracy and sampled at 44,100 Hz (you can obtain this number when you read the audio file with *audioread*). Modify your Matlab code so that your signal vector is read from a file, and you also read the correct sampling rate. Some example code follows:

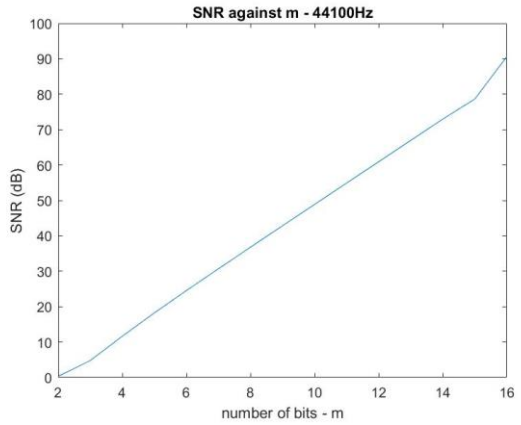
```
FilenameString=['ShortClip_Shine On.wav'];
[signal_vec,SampleFrequency_Hz]=audioread(FilenameString);
[Num_Samples,x]=size(signal_vec);
if x>1
    disp('Warning: This is not a mono recording')
end
Duration_sec = Num_Samples/SampleFrequency_Hz;
Display_String=[num2str(Duration_sec),' second audio file, sampled at
',num2str(SampleFrequency_Hz),' Hz'];
disp(Display_String);
```

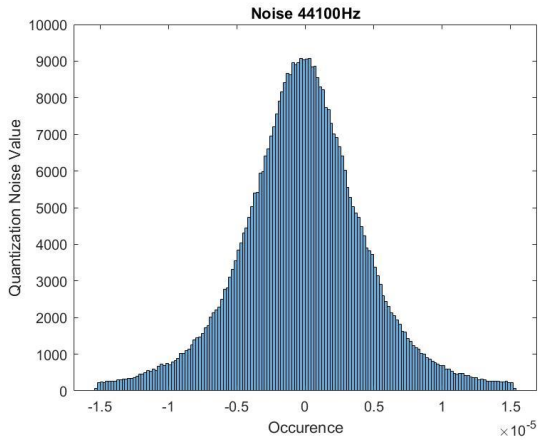
b) Again, measure SNR as a function of *m*, and listen to the quantised data using *sound*.

Is the slope of the SNR vs *m* function the same as in the case of the sinusoidal test signal? Up to how many bits of quantisation can you hear the quantisation noise? Is it easier to hear with speech or music?









The slope of the SNR vs m function is the same as in the case of the sinusoidal test signal.
The quantisation noise can be heard up to 8 bits for speech. After that, we can hear the original sample with clarity.
The quantisation noise can be heard up to 7 bits for music. After that, we can hear the original sample with clarity.
It will be easier to hear with music.

Were you correct in your speculation at the end of Section 5 above? That is, do you need to be so careful about the choice of sample rate? (Make sure you can explain why.)

Yes, we need to be careful about the choice of the sample rate. It would be similar but more noise is present.

Why?
The sampling frequency has to be twice the bandwidth of the signal to prevent aliasing and have to be reconstructed so that it will be ideal to the original signal.

7. Non-uniform Quantisation (Optional)

("Optional" does not mean "I stop here"! In terms of marks, you will not be penalised if you don't do this section, but you may be able to make up for some earlier lost marks if you do. More importantly, if you work through the steps below, you will get a more solid understanding of how and why non-uniform quantisation is used in many important applications – such as every telephone call you make! The section is only marked "optional" because if things don't all go smoothly earlier, you may struggle to get through it in the 2 hours lab time. If you don't/can't do it during the lab time, it would be a good idea to complete it in your own time afterwards.)

With non-uniform quantisation, the step size Δ is not the same for all input values, but instead the step sizes are small for small input values and larger for large input values. In this section, we will explore the value of non-uniform quantisation.

a) Repeat your quantisation test, but this time reduce the signal amplitude by a suitable factor (divide all the values of the signal vector by, say, 8 or 10) before quantisation. After you've quantised the signal and measured the SNR, scale all the values up again before playing the audio out, so that the volume is the same as for the original signal. Use *m=8* for this test.

Is the quantisation noise more, or less, noticeable for a smaller amplitude signal? Why? Measure the SNR, compare it with the result without the signal attenuation and try to explain the quantitative difference. (Hint: make sure you calculate SNR using the scaled down version of the signal, before you scale it up again to listen to; and make sure you understand why.)

Quantisation noise is more noticeable for the signal with smaller amplitude. The attenuated signal will have a lower SNR. The SNR value when m = 8 is 18.70 dB when attenuated and 36.78 dB when not attenuated. As a result, the signal amplitude will be reduced and the noise will increase.

Explain the relevance of this observation to the nonlinear (A-law or μ-law) quantisation that is used in digital telephony.

Based on the observation made above, we can see that a smaller amplitude will make the quantisation noise much more noticeable. When we only used 1/8 of the original quantisation while maintaining the same levels, the quantisation levels will drop. As a result, there will a be smaller step for the nonlinear quantisation case when we attenuated the signal by 8 times.

b) OK, if non-uniform quantisation is so good, let's test it. Matlab provides a function for 8-bit μ-law quantisation[12]. It is therefore easy to quantise a signal using this function, with the code:

```
mu=lin2mu(signal_vec);
Quant_vec = mu2lin(mu);
```

Repeat the test in part (a) above, quantising an attenuated version of the input signal, but this time use μ-law quantisation. Again, measure the SNR and listen to the result. (Note that you are only doing comparisons using 8-bit quantisation, because that's the only accuracy that the Matlab *lin2mu* and *mu2lin* functions operate with.)

What do you notice about the subjective quality of the quantised signal, compared with what you heard in part (a) above? What is the SNR, and how does it compare with the result from part (a)?

Compared to part a, the quantised signal has a better quality. The non-uniform quantisation has SNR value of 36.27dB while the 8-bit quantisation is only 18.78dB. Noise will be reduced. We can also deduce that the SNR of signal quantised with μ-law is higher than the signal quantized with uniform quantisation.

[12] It is sometimes referred to as "mu-law" to avoid the need to use the Greek character "μ".

Explain these observations.

Number of quantisation levels on the non-uniform quantisation consists of sufficient numbers along with minimum step size to have a clear difference between values for each sample. This will also increase the accuracy of the quantization as the quantization levels are not underutilized inefficiently which will ultimately lead to an improvement in the SNR value.

Explore the quantised representation. First, inspect the array "mu" (or whatever variable name you used in the code shown above) and see the range of numbers present, to convince yourself that this really is doing 8-bit quantisation. See if you can determine how many of the possible 8-bit numbers (i.e. 0-255) are actually used in the mu vector to represent the attenuated signal[13]. Compare this with the range of values used in the linear case, and also with the range of possible input values, and use this to explain the advantage of non-uniform quantisation.

The range of numbers present is from -1 to +1. Only approximately 1/8 of the possible 8-bit numbers are actually used in the mu vector to represent the attenuated signal.

As we can see, the non-uniform quantisation is more efficient compared with uniform quantization. This will lead to an increase in the accuracy quantization of the signal.

8. Conclusion

Quantisation is an absolutely fundamental and unavoidable component of any system that involves processing real-world analogue signals in digital form (including all image, video or audio capture systems, but also almost any other continuous quantity we measure today: temperature, pressure, weight, speed,…..). By completing this laboratory, you now have a feel for what the effects of quantisation are, and how many bits (or what SNR) are needed to avoid perceptible quantisation noise in the case of audio. You will also now understand some of the important design considerations for quantisers (midtread or midrise, uniform or non-uniform).

Before finishing, could each student please click on the "Feedback" icon for this laboratory on Moodle (all inputs are anonymous). Record the SNR needed before you felt that the quantisation noise was no longer audible in your audio tests on the sinusoidal test tone, the speech sample, and music (5(d) and 6(b) above). This will allow us to obtain an overall

[13] Because negative numbers are stored in ones complement, this is not just a matter of using min() and max() functions. One way is to use histograms and find how many numbers in the range 0-255 are unrepresented in mu. Or you may have a more elegant solution?

average, with which you can compare your result. There is also an opportunity to provide some brief feedback on this laboratory exercise.

Finally, please submit this completed report via Moodle by the stated deadline. In so doing, please be aware of the following:

- Even if you have had a mark assigned to you during the lab session, this mark will not be registered unless you have also submitted the report.
- Your mark may also not be accepted or may be modified if your report is incomplete or is identical to that of another student.
- By uploading the report, you are agreeing with the following student statement on collusion and plagiarism, and must be aware of the possible consequences.

Student statement:

I have read the University's statement on cheating and plagiarism, as described in the *Student Resource Guide*. This work is original and has not previously been submitted as part of another unit/subject/course. I have taken proper care safeguarding this work and made all reasonable effort to make sure it could not be copied. I understand the consequences for engaging in plagiarism as described in *Statute 4.1 Part III – Academic Misconduct*. **I certify that I have not plagiarised the work of others or engaged in collusion when preparing this submission.**

– END –