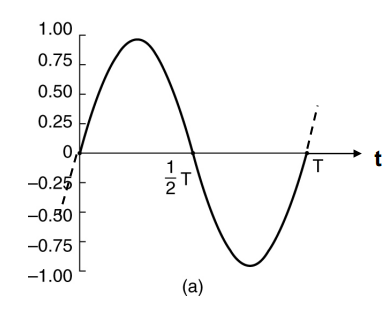
Real world signals include sound (speech, music), vision (photos, natural scenes), radio transmissions etc. All arrive as physical quantities that vary continuously and can carry an infinite amount of information.

*How can we represent and process these digitally?*



For a sine-wave of amplitude **1**, and a period of **T** seconds:

Freq is 1/T Hz.

A sine wave is:

* continuous in time, a value can be measured at any point in time
* and continuous in value, takes all real values between

**Sampling an analogue signal**

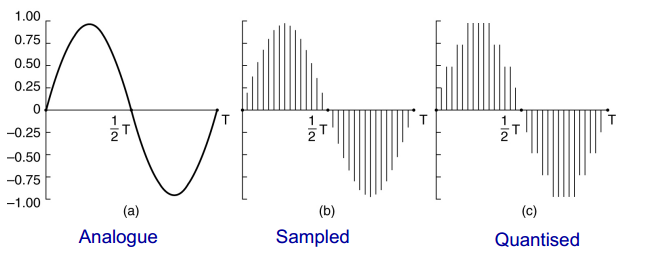
We take samples by measuring the value of the signal at regular points in time.

It now becomes a discrete (rather than continuous) time signal, but the amplitude remains continuous.

What do we lose by doing this? Answer: *Nothing provided we sample frequently enough.*

**If the sampling freq >= twice the maximum freq we wish to capture, nothing is lost.**

If we know the sampled values exactly, we can always recreate the original analogue signal exactly from these samples. This is know as the [‘Sampling Theorem’ or ‘Nyquist criterion’.](https://www.cs.cf.ac.uk/Dave/Multimedia/node149.html)



**Quantisation**

There are a limited number of bits for storing each sample. So each sample becomes quantised, and cannot represent the full range of continuous values of an analogue signal.

The Sine-wave representation will be a series of integers.

Quantisation introduces errors in sample values. With many bits per sample (eg, 16) the error will be small, but this requires high storage or transmission capacity (bits per sample - quantisation error trade-off).

If we reduce the number of bits per sample, the error becomes larger & noticeable.

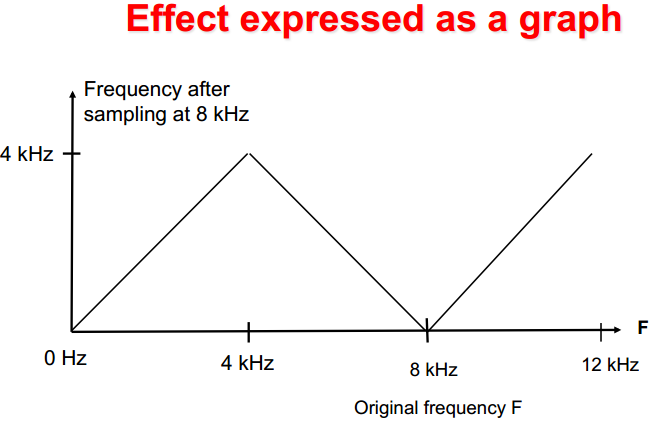
In sound recordings we hear it as ‘quantisation noise’, in images, it is seen as a quality degradation.

**Aliasing**

A sampling freq of *Fs* Hz is good for input freq up to *Fs/2* Hz.

For higher frequencies, we get aliasing. For a sine-wave of freq *F > Fs/2,* after sampling it becomes a sine-wave of freq *Fs - F,* assuming Fs/2 < F < Fs.

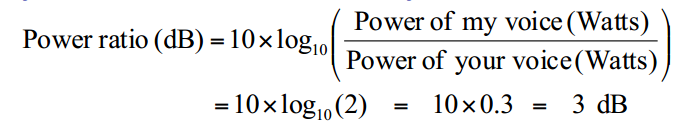
eg. If Fs = 8kHz, F = 5kHz, F will become 3kHz.



Further examples in the lecture notes, quite self-explanatory

**Decibel Scale**

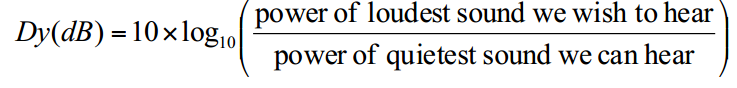
The **decibel** (**dB**) is a [logarithmic unit](https://en.wikipedia.org/wiki/Logarithmic_unit) used to express the ratio between two values of a physical quantity, often [power](https://en.wikipedia.org/wiki/Power_%28physics%29) or [intensity](https://en.wikipedia.org/wiki/Intensity_%28physics%29). In this case we are comparing sound ‘loudness’.



**CD quality digital audio**

Humans can hear sound over a freq range of 20Hz to 20kHz, therefore CD sampling frequency is 44.1kHz (an extra 4.1kHz added on just in case).

Dynamic range of Human hearing:



Which is about 120 dB, a power ratio of.

So how many bits/sample do we need?

For uniform quantisation we get around 6 dB per bit. For 120 dB we need 20 bits, too many for when CDs were invented.

We settle for 16 bits per sample and the use of [dynamic range compression](http://en.wikipedia.org/wiki/Dynamic_range_compression).

So therefore a CD data rate (stereo) = 16 x 44100 x 2 = 1411 kbit/s.

**Telephone quality digital speech**

Uses Narrow-band freq of 50/300Hz - 3.4kHz.

Loses ‘naturalness’ but not intelligibility. Not incredibly clear but understandable.

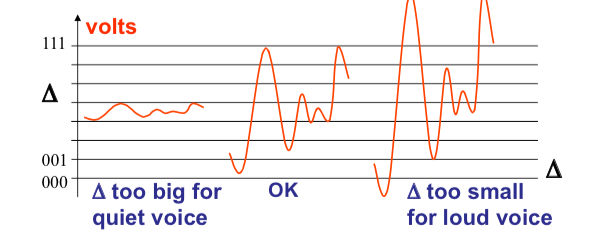
Sampled at 8kHz with 8 bits per sample, leading to a 64 kbit/s bit-rate. Uses mu-law or A-law *non-uniform* quantisation. (ITU-G711 standard)

**Uniform quantisation**

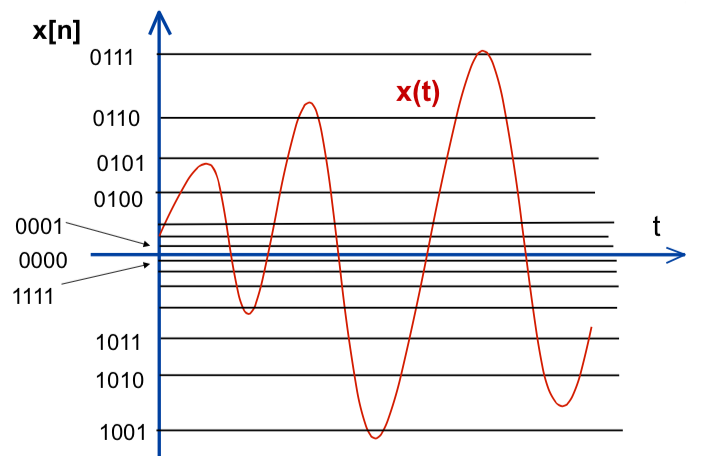
Each sample of speech at x(t) is represented by a binary number x[n].

Each binary number represents a voltage. There is a constant voltage difference between the voltages for adjacent binary numbers, eg between 0001 and 0111 (4-bit). We call the quantisation step-size.

The problem with uniform quantisation is choosing the right value of . One solution is to adjust according to amplitude of sample. Make the step-size change from sample-to-sample.



**Non-uniform quantisation** involves non-constant step-sizes, so each binary number will represent a different range of amplitude values.



**Implementation**

1. Digitise x(t) accurately with *uniform* quantisation to give x[n].
2. **Compander** increases smaller amplitudes of x[n] and reduces larger ones, to give y[n].
3. Uniformly quantise y[n] using fewer bits.
4. Store or transmit the companding result, pass it through an **expander** to reverse the compander effect (decrease small, increase large).

Companding is like ‘compression’, but it is done for coding purposes, not for listening to directly. Famous companding formulas - A-law & Mu-law (G711).

**Quantisation error and noise**

Uniform quantisation produces error in each sample, random in the range of .

When samples are converted back into analogue form, error is heard as white noise, an unwanted signal, which is spread evenly across all frequencies.

The **Mean square value[[1]](#footnote-0)** (MSV) of noise is: **/12.**

The **Mean square value** of amplitude is A^2/2

From this we can derive the **Signal-to-quantisation noise ratio** (SQNR) which reflects the relationship between the maximum signal strength and quantisation error that is introduced. Applies only to *uniform quantisation.*

*SQNR is in dB, so*

SQNR in terms of decibels is **6 dB per bit + 1.8.**

Examples

* 8 bits/sample, with a fixed and uniform quantisation. The SQNR for loudest talkers is 6 x 8 + 1.8 = 49.8 dB.
* For CD recordings with 16 bits/sample, the max SQNR will be 6x16 + 1.8 = 97.8 dB. This is for the loudest music. For quieter music we will be able to hear quantisation noise.

Remember that dB is a measurement of comparison, so the larger the SQNR, the larger the ratio between wanted signals and unwanted noise.

**Speech/music for mobiles**

64 kbits/second is still too high for mobile telephony, we aim to encode speech at around 13 kb/s or lower (as was done in lab 1).

1. Basically the square root of the mean of the squares of a sample. [↑](#footnote-ref-0)