Module IN3031 / INM378 Digital Signal Processing and Audio Programming

Tillman Weyde
t.e.weyde@city.ac.uk

Digital Signals: Sampling and Quantisation

Signals in the Time Domain

- **signals** (one channel)
  - analogue $x_a(t): \mathbb{R} \to \mathbb{R}$
  - digital $x_d[n]: \mathbb{Z} \to \mathbb{Z}$
- $x_d[n] = x_a(n \cdot 1/Fs)$ where **Fs** is the **Sampling Frequency**

Spatial Signals: Images

- **signals** (one channel)
  - analogue $x_a(x, y) : \mathbb{R}^2 \rightarrow \mathbb{R}$
  - digital $x_d[n, m] : \mathbb{Z}^2 \rightarrow \mathbb{Z}$
- $x_d[n,m] = x_a(n \cdot 1/Fs, m \cdot 1/Fs)$ where **Fs** is the <u>**Sampling Frequency**</u>
- We focus on digital signals from here on

Operations in the Time or Space Domain

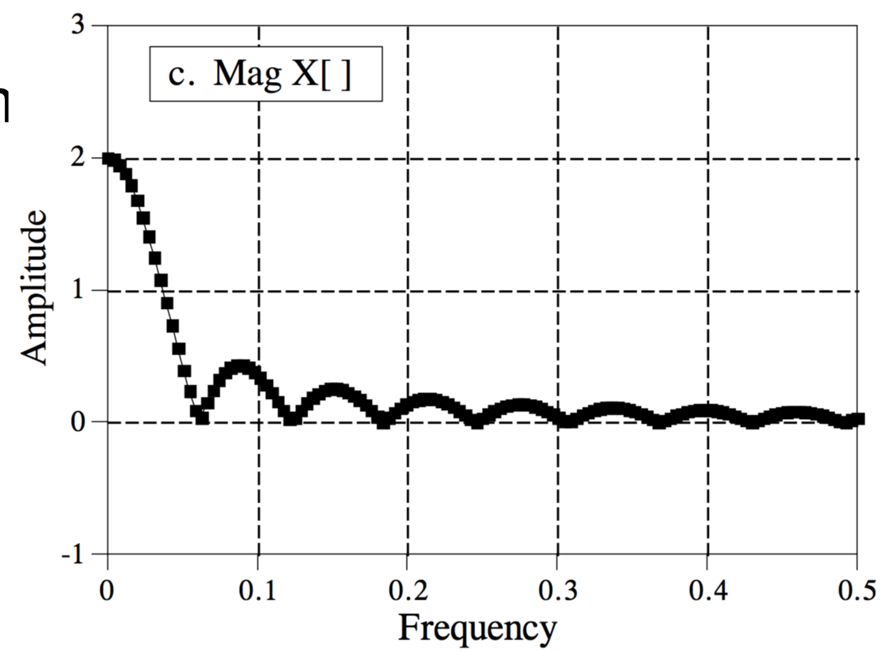
- changing amplitude = **multiply**ing with a number *a*
  $y = a \cdot x$, i.e. $y[n] = a \cdot x[n]$
  $|a| > 1$: louder/brighter signals, $|a| < 1$ softer/darker signal

- mixing signals = **add**ition
  $y = x_1 + x_2$, i.e. $y[n] = x_1[n] + x_2[n]$

- delay = time-**shift**ing
  $y[n] = x[n-k]$

Frequencies and Spectra

- **Most signals** contain **multiple frequencies** (harmonic, inharmonic, noise ...)
- **Amplitude** of the signal **per frequency** is called the **spectrum**
- The **square** of the spectrum is the **power spectrum**
- We will address **later** how to **calculate the spectrum**



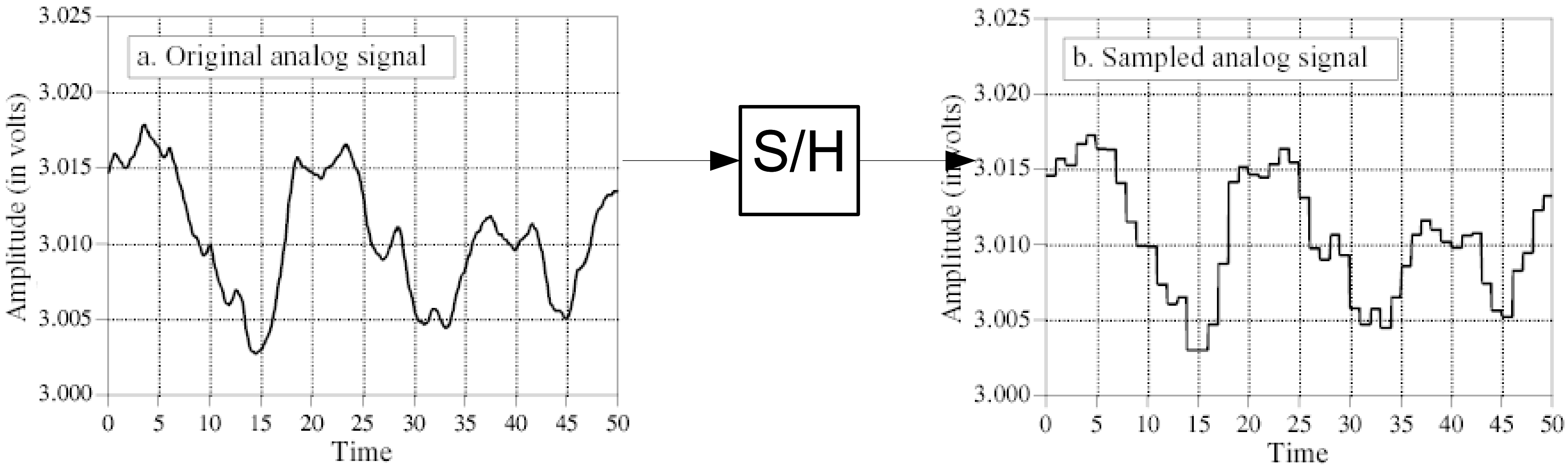
Example: Digital Sound

- **Analogue** systems use **continuous** values
- **Digital** systems use **discrete** (non-continuous) values

- **Digitisation** reduces from **continuous to discrete**:
  - **time** (by **sampling**)
  - **amplitude** (by **quantisation**)

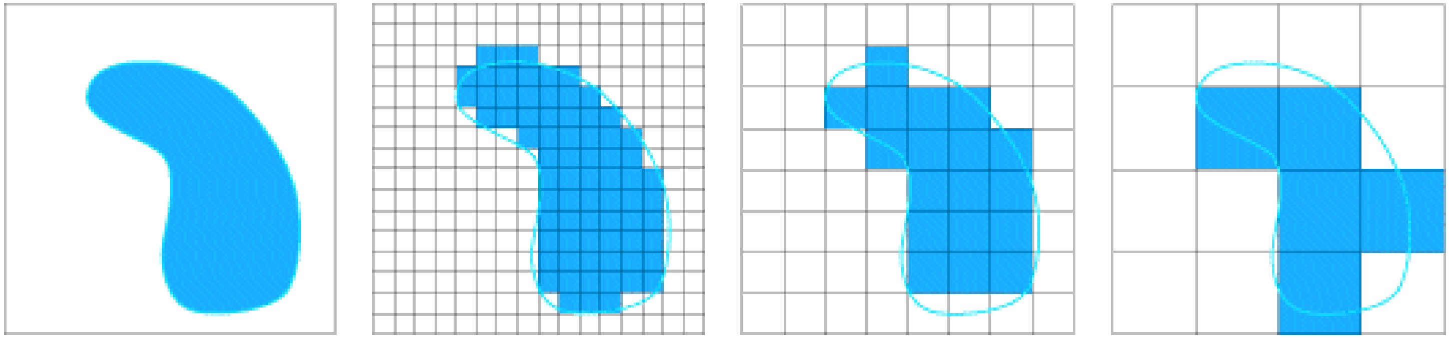
Digitising Time: Sampling

- **Sample/Hold** electronics:
take a **value** at **regular time intervals** and **hold** it
- ***Sampling Frequency*** *(**sampe rate**, often **Fs** or **fs**):*
  Number of **samples per time**, i.e. **time resolution**



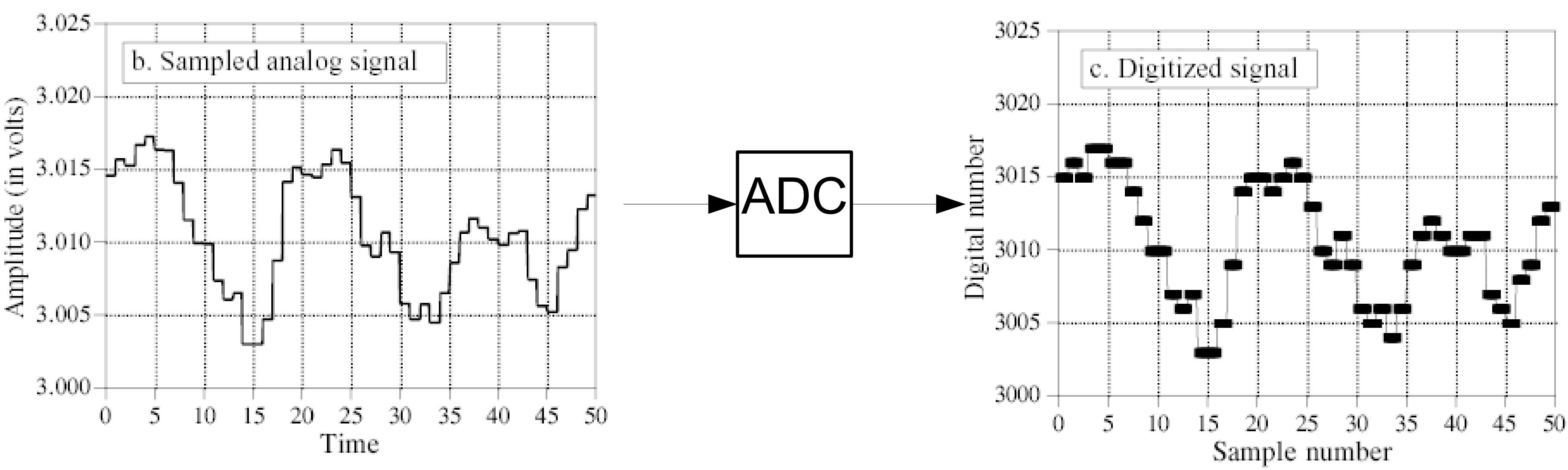
Digital Images: Spatial Sampling

- **Spatial resolution (raster size) - spatial sampling frequency**
- **Sample resolution per dimension,** often in dots per inch (DPI)
- **Typically same** resolution in both **dimensions**



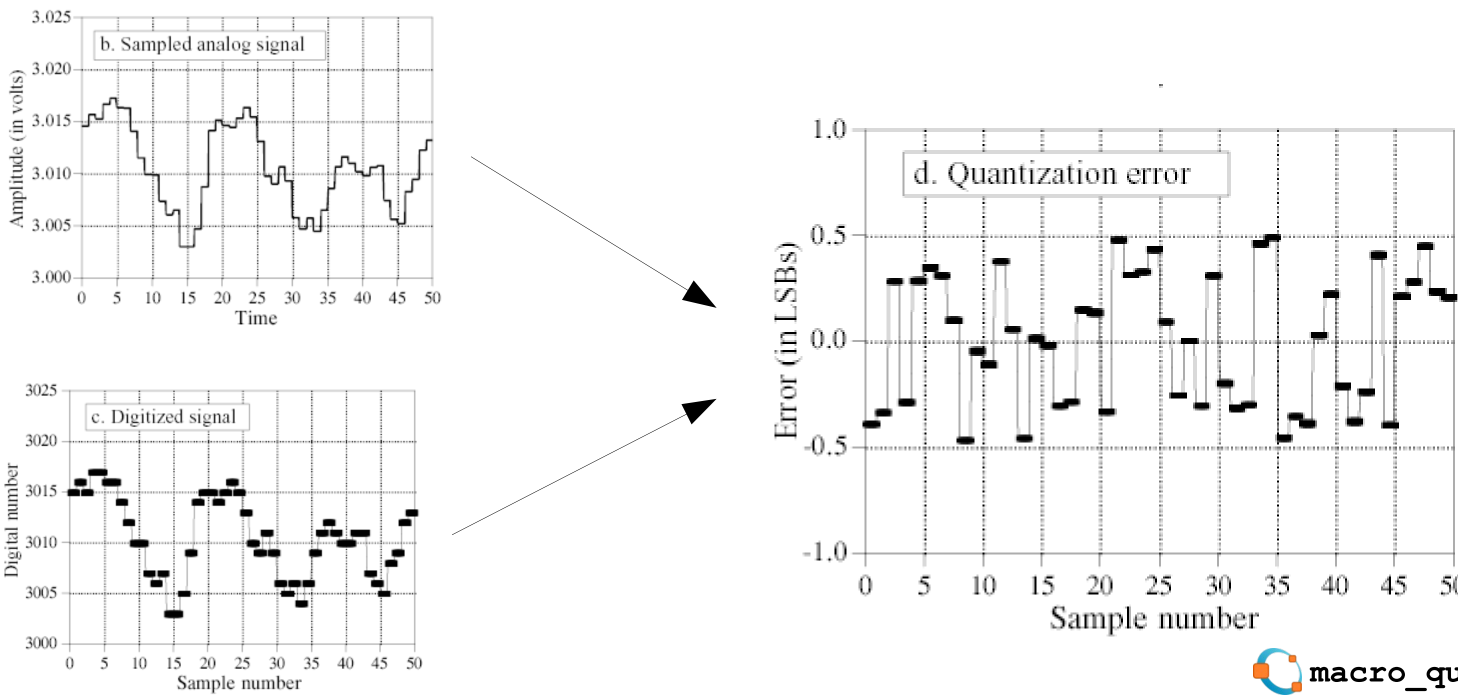
Digitising Values: Quantisation

- Rounding a continuous to a discrete value (from a fixed set)
- **Sample Resolution** (Depth): **number of bits per sample**
  - Defines the possible range of values
  - e.g. 8 bits (2^8=256), 16 bits (2^{16},~65k), 24 bits (2^{24},~16m)



Quantisation Error

- **Difference** between the **sampled** and **quantised** signal (rounding error)
- Different values are mapped to one -> **information loss**.



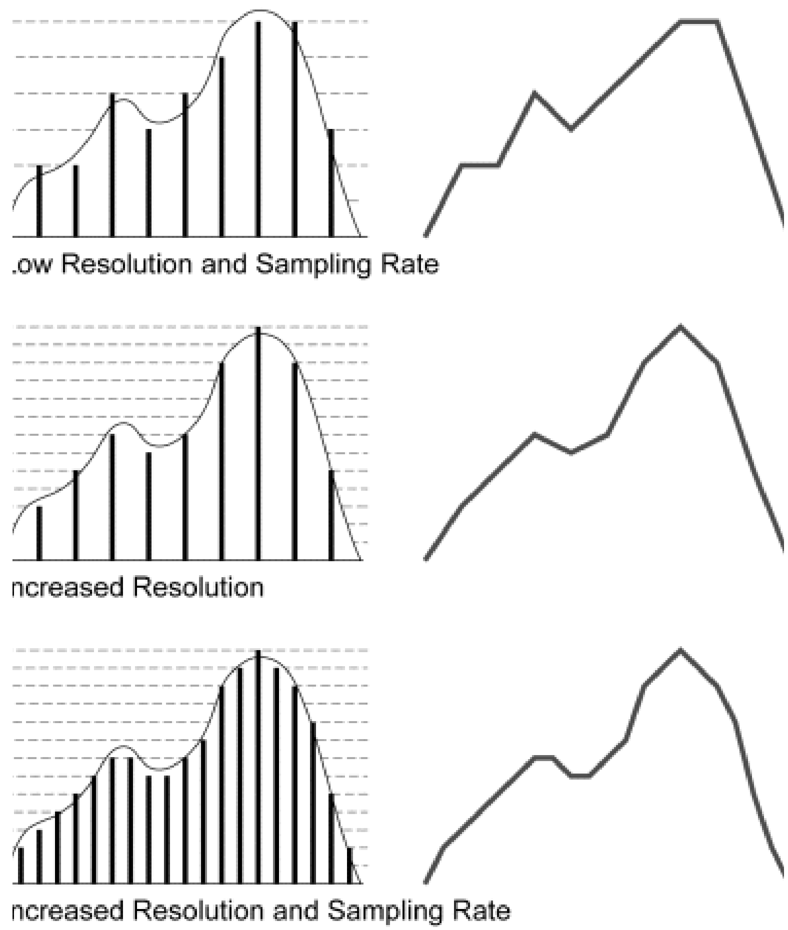
macro_quantisation

Sampling, Quantisation and Signal Quality

Sampling rate: time resolution

Bit depth: value resolution

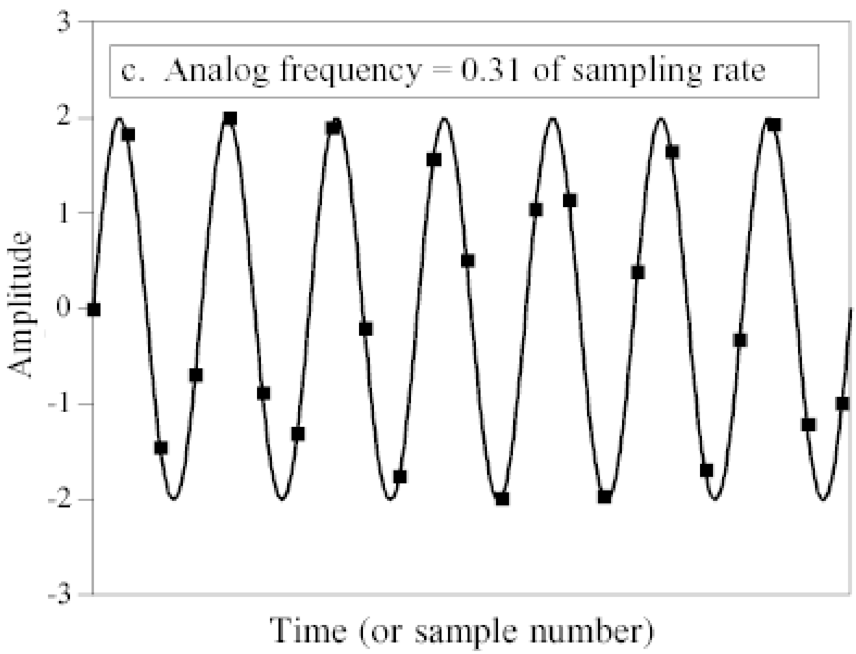
- **Higher resolution**: **lower quantisation error** (closer to the original)
- **Crucial** for signal **quality**



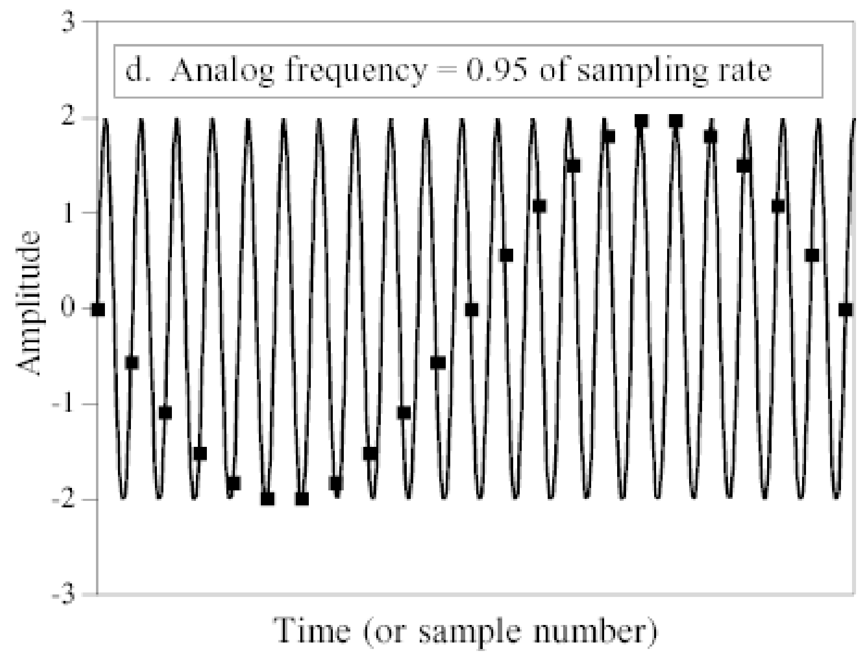
Sampling and frequency

- a problem with **high input frequencies** relative to **Fs**
  sampled signal looks quite different from input

low input frequency



high input frequency



Aliasing

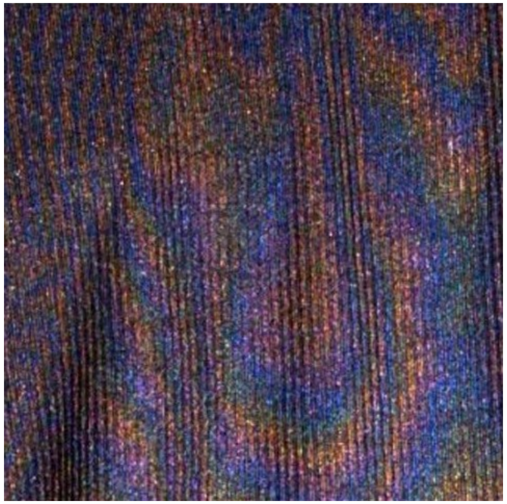
- Intuition: **2 samples** needed per **wave cycle** (one for each peak and trough)

- **Output frequencies** are different (aliased) **too low** if too **few samples** i.e. temporal/spatial resolution is too low

macro_sampling

Spatial Aliasing (Moire)

- In **2D**, the **same problem** occurs
- **E.g. woven patterns** can exceed **camera resolution**
- Effect can be **different per colour channel**



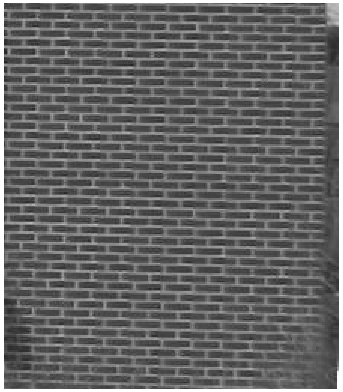
‘Digital’ Aliasing

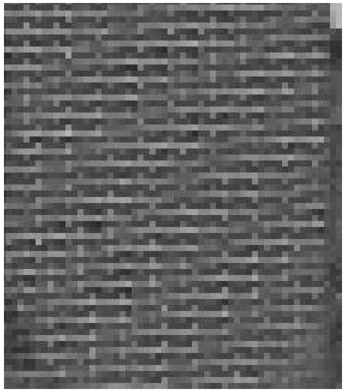
- Aliasing occurs **not only** when sampling **physical signals**.

- In in the **digital domain aliasing** can occur by:

  - **Downsampling** digital signals (reducing resolution)

  - **Sampling mathematical functions** (synthesizing signals)

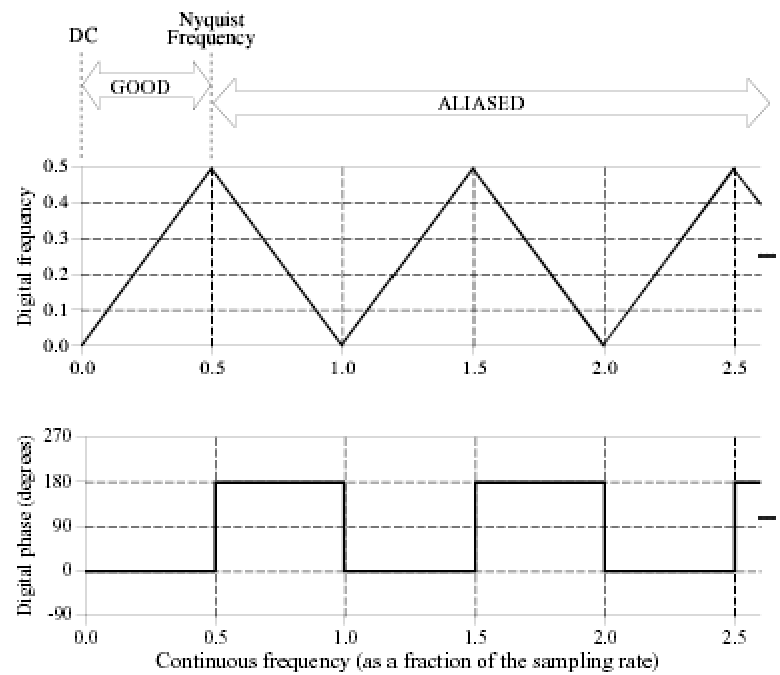




Sampling Theorem

- **Sampling cannot capture frequencies greater than half the sampling frequency**
every wave cycle needs two points

- **Fs/2** called ***Nyquist-Frequency***
- Frequencies in the signal above the *Nyquist-Frequency* get **mirrored down** at the *Nyquist-Frequency* (***Aliasing***)
- $f_{al} = -abs([f_i \ mod \ F_s] - F_s/2) + F_s/2$



Filters

- **Filters -** signal processing units (typically) designed to **remove frequency components**
- **Filter types** named mostly by **frequency ranges** (**bands**) that can **pass through** the filter, e.g.

  high pass

  low pass

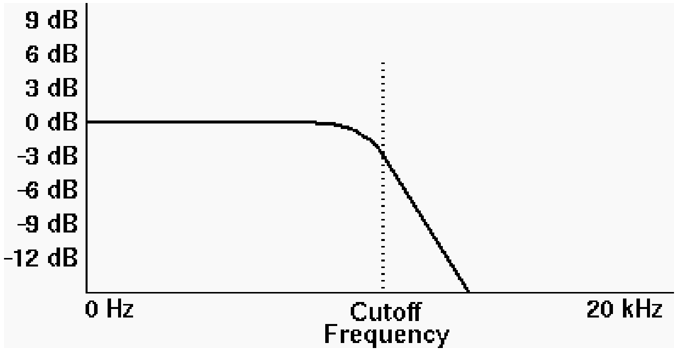
  band pass

  band rejct
- Typical examples are **EQ** in stereos and mobile phones

Aliasing Solution

- **Increase** time or space **resolution**
  not always possible/practical
  may not (fully) resolve the problem

- **Anti-alias filter**:
  Remove components
  above the **Nyquist frequency**
  before (down-)**sampling**
  (with a low-pass filter)

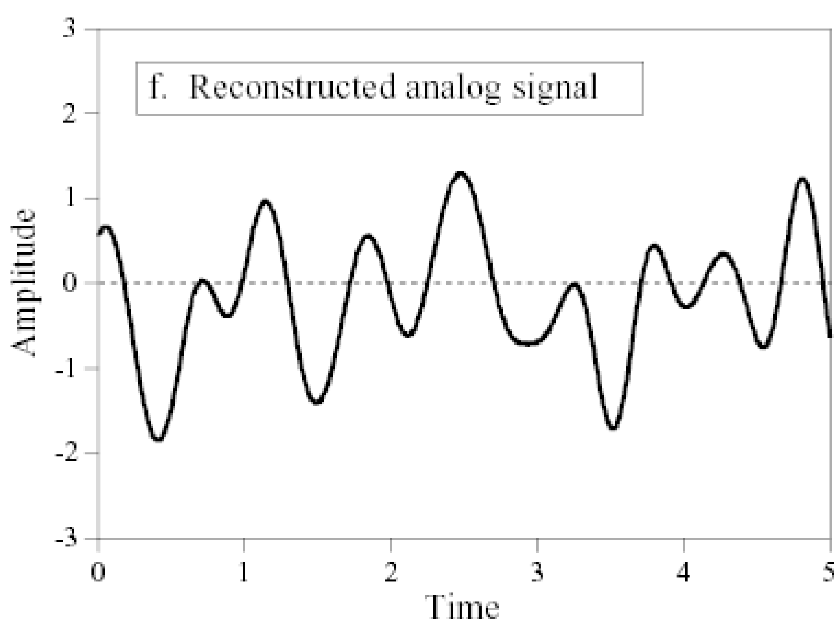


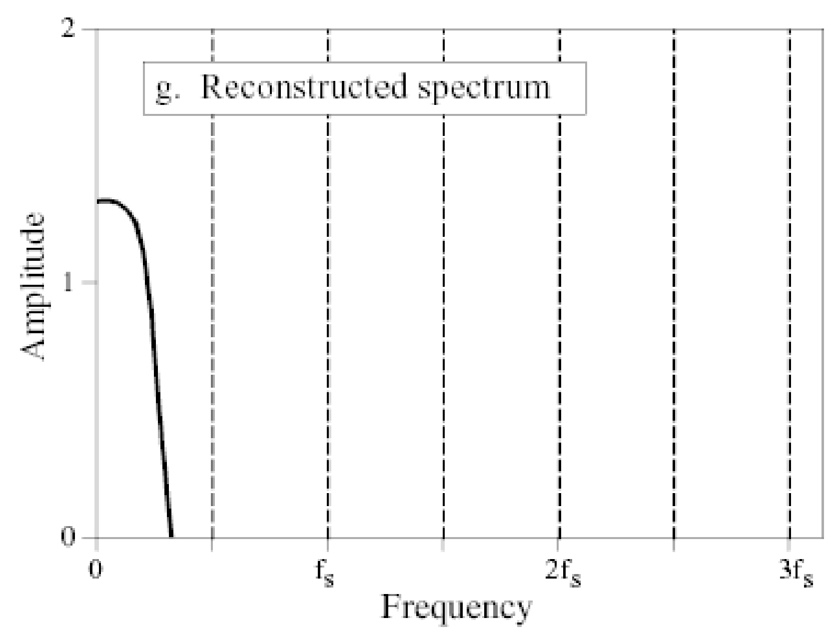
Reconstruction of a Sampled Signal

- Goal: reconstruction of the **original signal** (within the limits of the sampling theorem)
- **Problem**: samples provide **discrete values** that we need to be **connect continuously**
- Reconstructed signal should have the **same frequency content as the original**

Signal Reconstruction

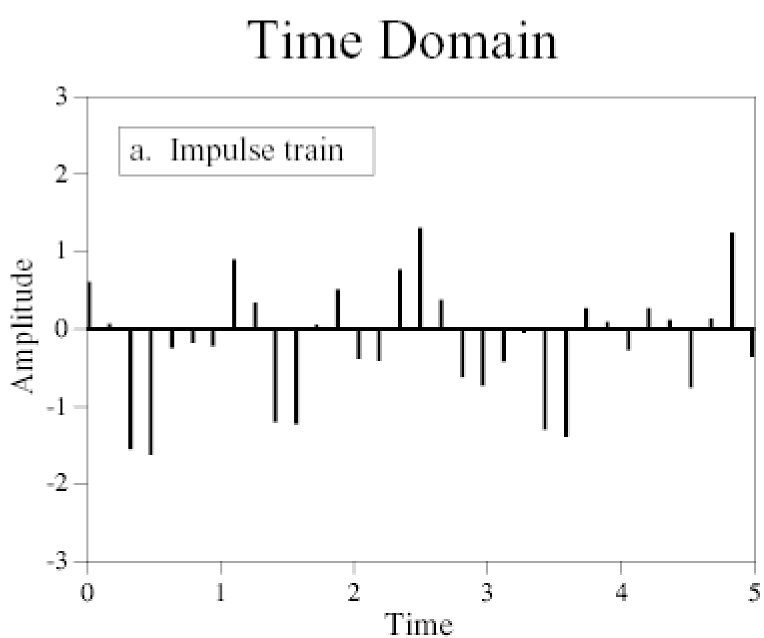
- Reconstruction should reproduce the original signal and spectrum

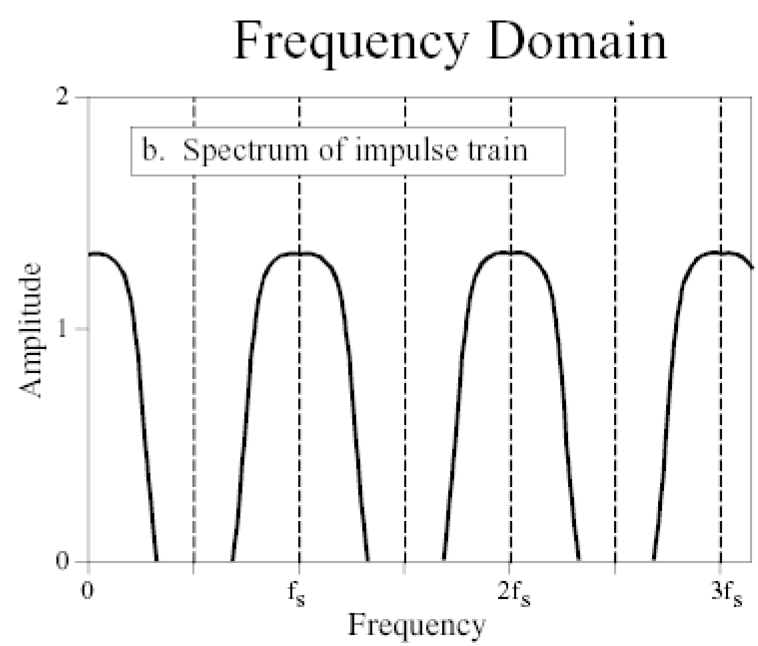




Signal Reconstruction

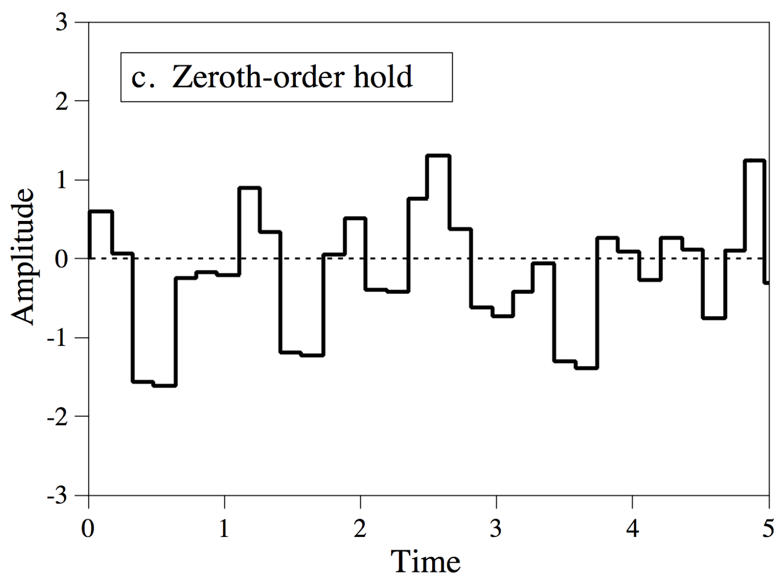
- **Ideal impulses** contain infinite frequency content, which **repeats at Fs multiples**.
- Easy to **filter** (analogue) but **not practical** to **generate**

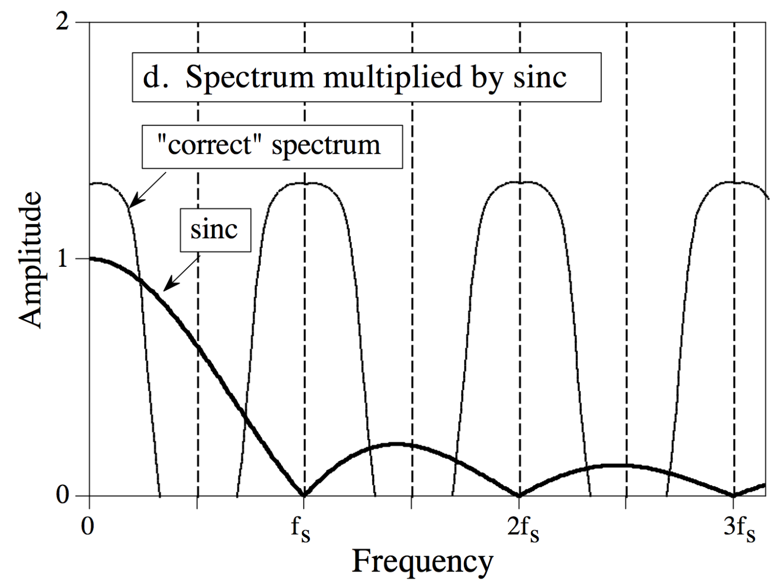




Signal Reconstruction

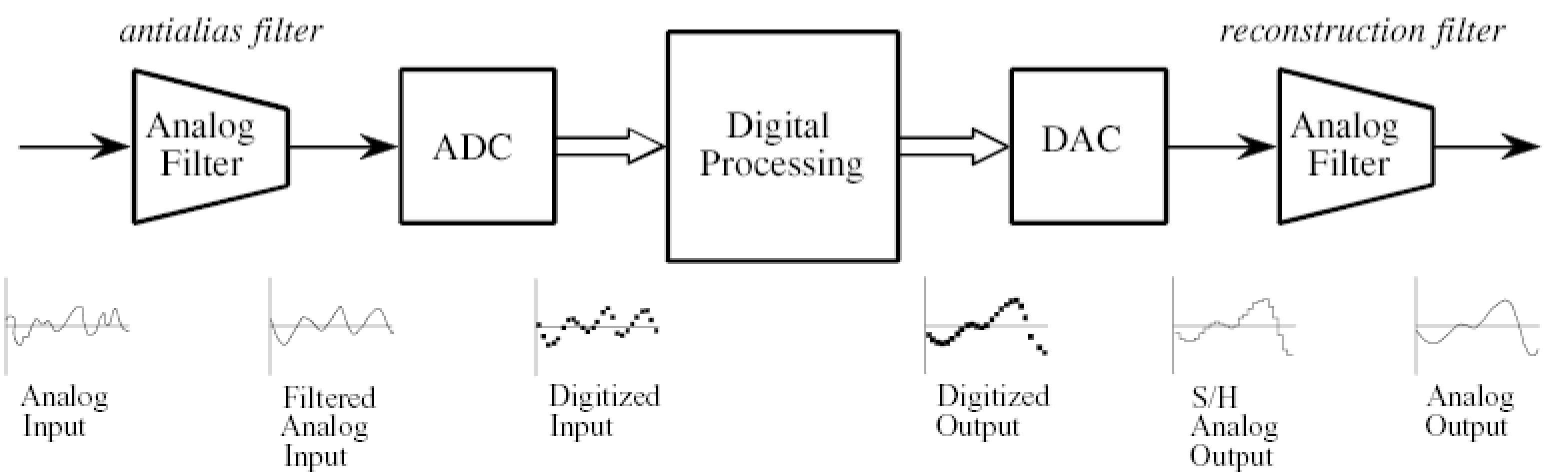
- Signal reconstruction by **holding the value effectively multiplies** the **spectrum with a sinc function** (sin(x) / x), better but still not ideal.
- **Further filtering** is needed, more in the next weeks.





Filtering in the ADA Chain

- Analog **input** must not contain frequencies higher than Nyquist-F.
  - **anti-alias filtering** (low-pass)
- **Output** created from digital contains additional frequencies
  - **reconstruction filtering** (low-pass)



Generating Signals

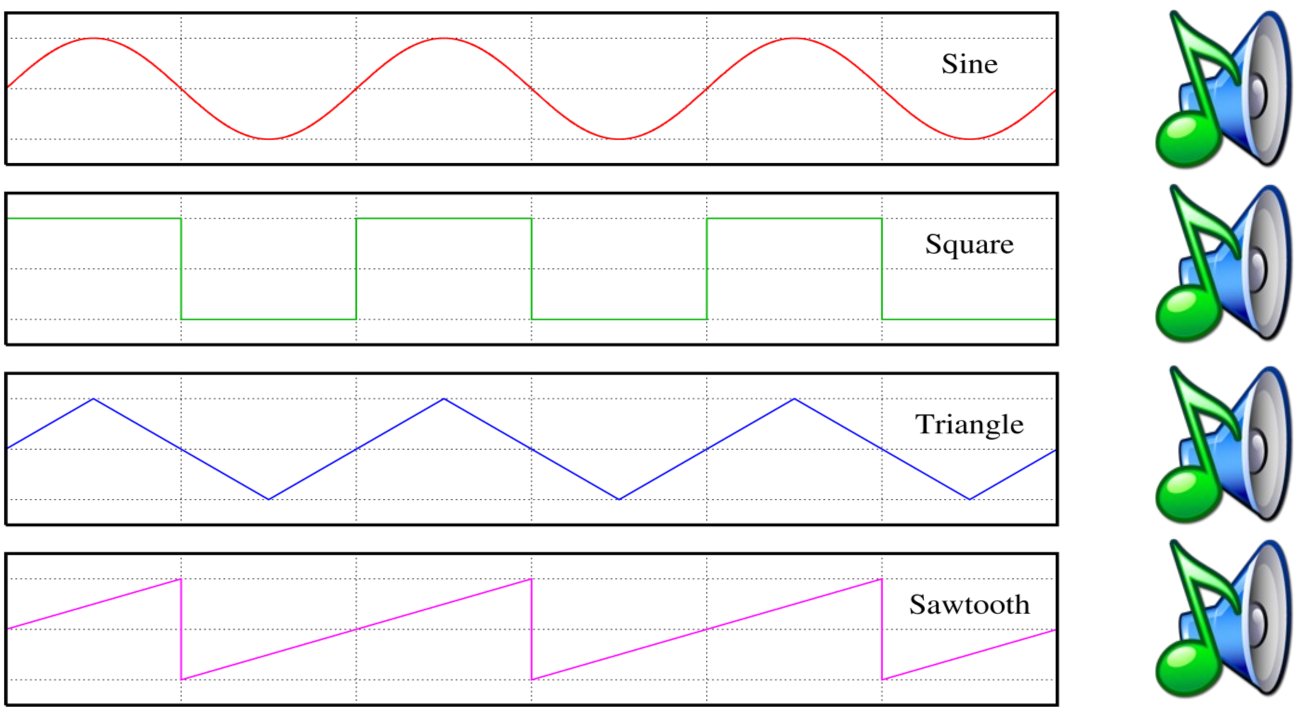
- **Generating** a signal can be done
  - **off line** or in **real time**
  - **digital** or **analogue**
  - **(re-)using signal waveforms**
    - **simple periodic**
    - **noise** (random)
    - **recorded** signal

Oscillators

- **Simple signal generators**
- **Periodic** waveforms (typical)
  - **sine**
  - **square**
  - **pulse**
  - **sawtooth**
- **Noise**: different ‘colours’

Oscillator Waveforms

- Periodic waveforms



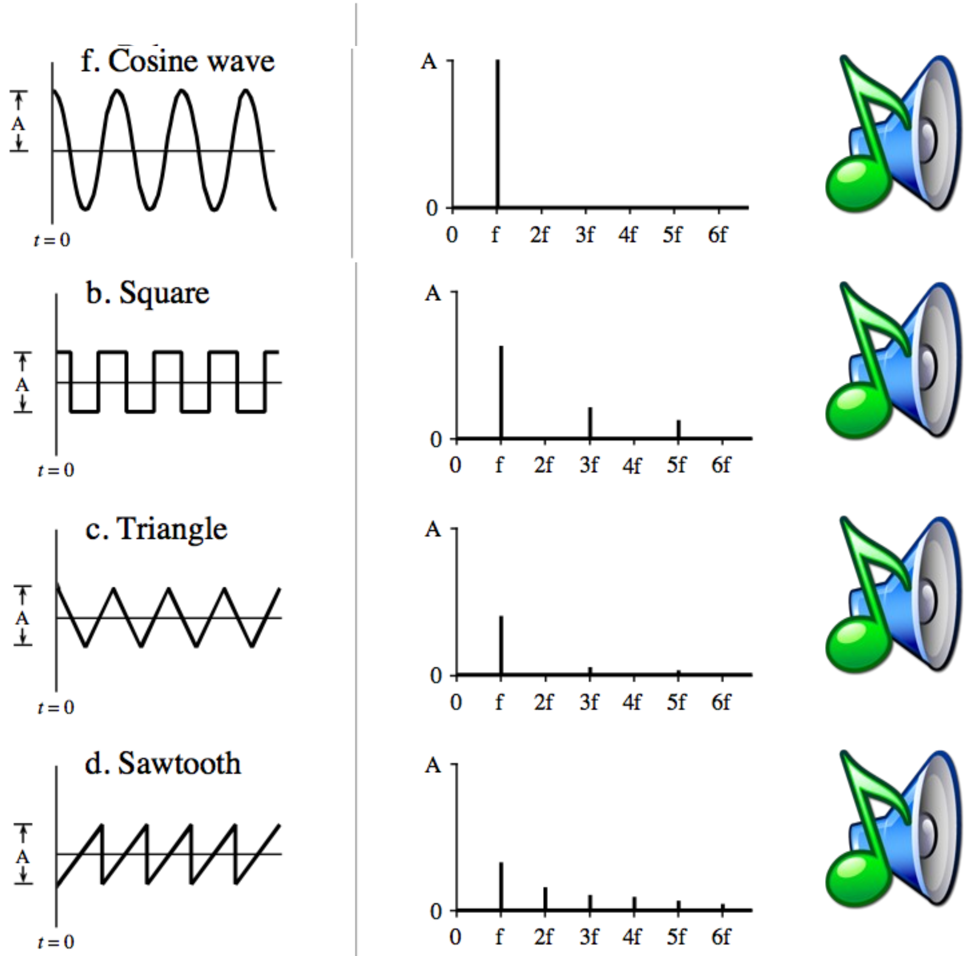
macro_signal_gen

Frequencies and Waveforms

- Simple periodic waveforms create **harmonic signals**

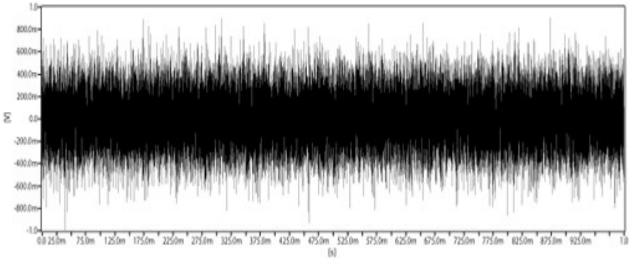
  (frequency components at integer multiples)
- How can we determine the frequency content of a given signal?

  We'll see next week :-)



Noise Oscillators

- Noise oscillators can create **random signals**
  - **white noise** has evenly distributed frequency components
  - **pink noise** has weaker high-frequency components (amplitude ~ 1/f).
- Some rarely used forms of noise
  - **brown** noise ($1/f^2$)
  - **blue** noise (f)
  - **violet** noise (f^2)



Control

- Most components of a synthesiser have some **parameters** to control
- In analogue systems electric control signals were used:
  - voltage controlled **oscillator**
  - voltage controlled **filter**
  - voltage controlled **amplifier**
- **External control** sources can be a musician playing on a keyboard, nowadays done in MIDI
- **Internal control** sources are low frequency oscillator (**LFO**) and envelope generator (**ADSR**)

Amplitude Control (Gain)

- In a computer, a gain control unit **multiplies** every sample with a **gain factor**:

  $y[t] = x[t] * c$

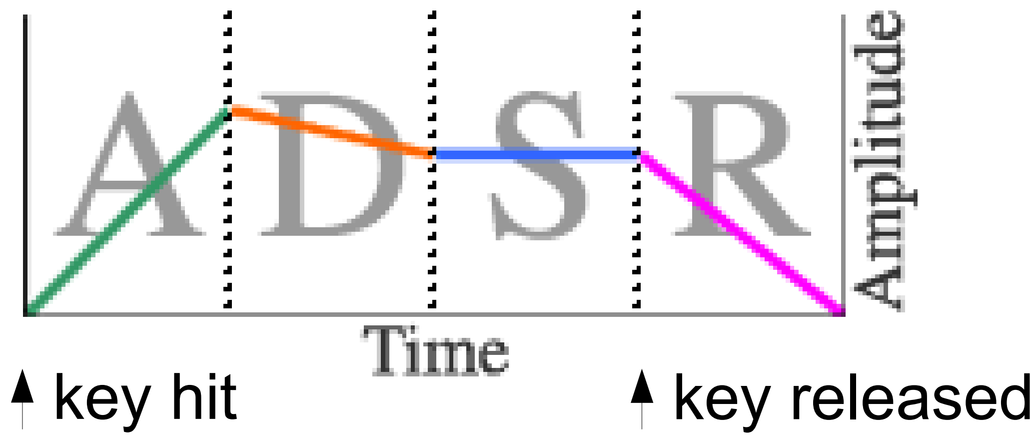
  in Matlab **`y = x .* c`** (the '.' is optional)
- *c* can **change over time**, in that case the unit is called **time variant**

Envelope Generator

- Natural sounds **change over time**
- This is modelled by an **envelope generator** that can **control gain** and other **system parameters**
- Envelopes are typically **triggered** by **events**
  - in music, typically a MIDI note-on/off message
  - in games, an event from the game play
- The **envelope** is **modulating signal properties** (e.g. amplitude, spectrum).

Envelope Generator (2)

- The most common form is an **Attack, Decay, Sustain, Release** (ADSR) generator.



Attack, Decay and Release have rate parameters, Sustain has a level parameter (usually not changed in real time).

macro_envelope

READING

http://www.dspguide.com/ Chapter 3

Maths refresher: Rochesso, D.: Introduction to Sound Processing Appendix pp. 154: Vectors and Matrices, Exponentials and Logarithms, Trigonometric Functions

NEXT WEEK:
Frequency Analysis