



FUNDAMENTALS OF IMAGE AND VIDEO PROCESSING

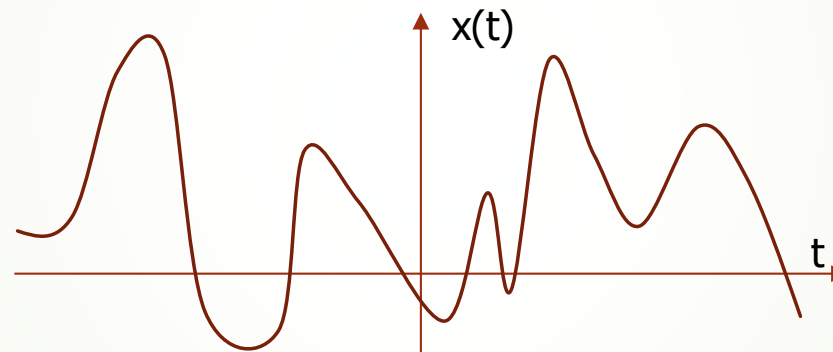
Part 2: From analog to digital signals

What we'll see in this section

- Analog vs digital signals
- Analog to digital conversion
 - Ideal sampling, Nyquist theorem and aliasing
 - Quantization, quantization noise
 - Bitrate
- Digital systems
 - Discrete convolution
 - Discrete signals in the frequency domain (DFT)
 - Filtering in digital frequency domain

Analog signals

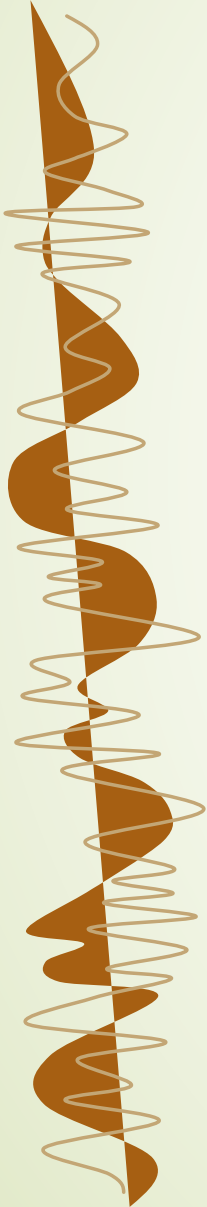
- Signals acquired by sensors are typically in analog form
 - Continuous in time/space (defined at every instant/point)
 - Continuous in amplitude (any value in a given range is admitted)



analog time signal

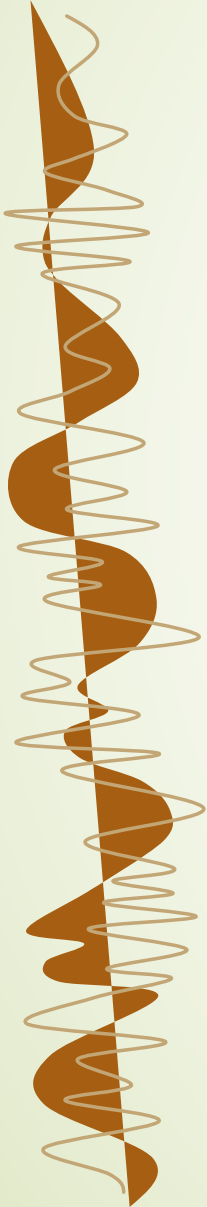
- This kind of signals cannot be managed directly by a computer, it is necessary to convert them in binary format

Analog vs digital



- Digital signals present several advantages with respect to their analog counterparts:
 - Easier processing → analog processing requires dedicated hw, digital processing can be implemented in software on a computer
 - Easier transmission → digital signals can be readily and more robustly transmitted over data networks (e.g., Internet)
 - Easier storage → digital signals can be compressed and stored on data repositories at extremely low cost
 - Easier interfacing/interoperability → digital data can be easily transferred, transcoded, exchanged among different systems/standards

Analog to digital conversion (ADC)

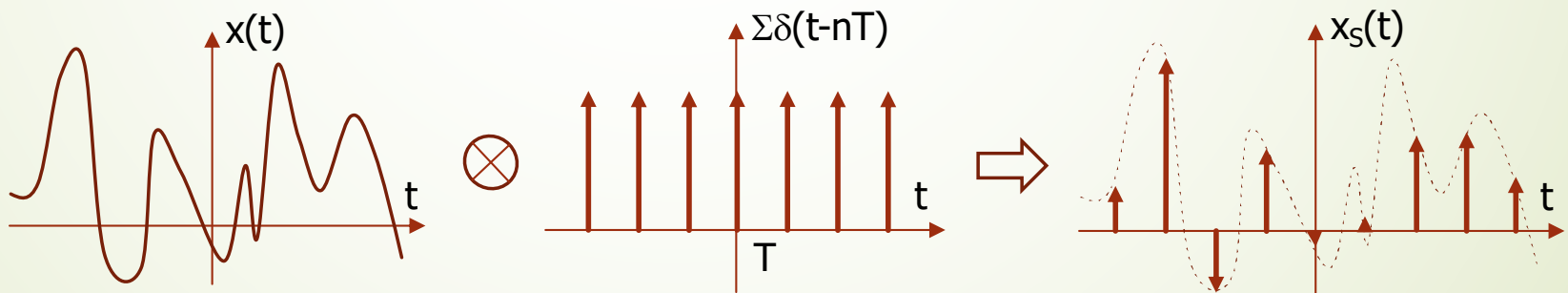


- An analog-to-digital converter is an equipment that transforms an analog signal into a numerical one
- It performs three main operations, in sequence:
 - Discretization in time(space) → **sampling**
 - Discretization in amplitude → **quantization**
 - Numerical representation of samples → **coding**
- In the following we'll provide some details on how to perform such operations, while preserving the content of the signal

Ideal sampling

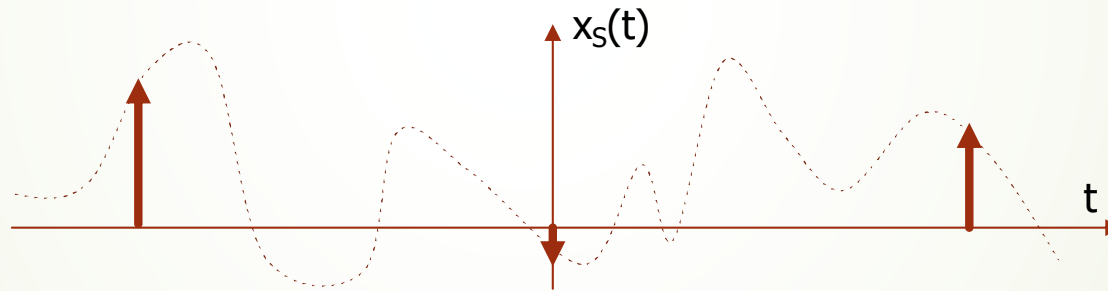
- Sampling can be thought as the operation of extracting a series of instantaneous values at specific points in the original signal
- Ideally, this operation can be implemented by multiplying the signal by a sequence of equally-spaced unit impulses
 - Each impulse captures the amplitude of the signal at a specific point and associates it to the area of the impulse itself

$$x_s(t) = x(t) \cdot \sum_{n=-\infty}^{+\infty} \delta(t - nT) = \sum_{n=-\infty}^{+\infty} x(nT) \cdot \delta(t - nT)$$



Sampling frequency

- The distance between two consecutive impulses T is called **sampling interval**
- More commonly, we use the inverse of T , which expresses the **sampling frequency** $f_s = 1/T$
- Intuitively, the sampling frequency cannot be completely arbitrary and independent of the signal characteristics



If the sampling frequency is too low with respect to the signal fluctuations, we cannot pretend to follow every signal variation

How could we determine the correct frequency?

Sampling frequency

- To derive the minimum sampling frequency, we need to look at the signal in the Fourier domain
- To this purpose we should introduce the FT of an impulse series (non demonstrated):

$$\mathcal{F}\left(\sum_{n=-\infty}^{+\infty} \delta(t - nT)\right) = \frac{1}{T} \sum_{k=-\infty}^{+\infty} \delta\left(f - \frac{k}{T}\right)$$

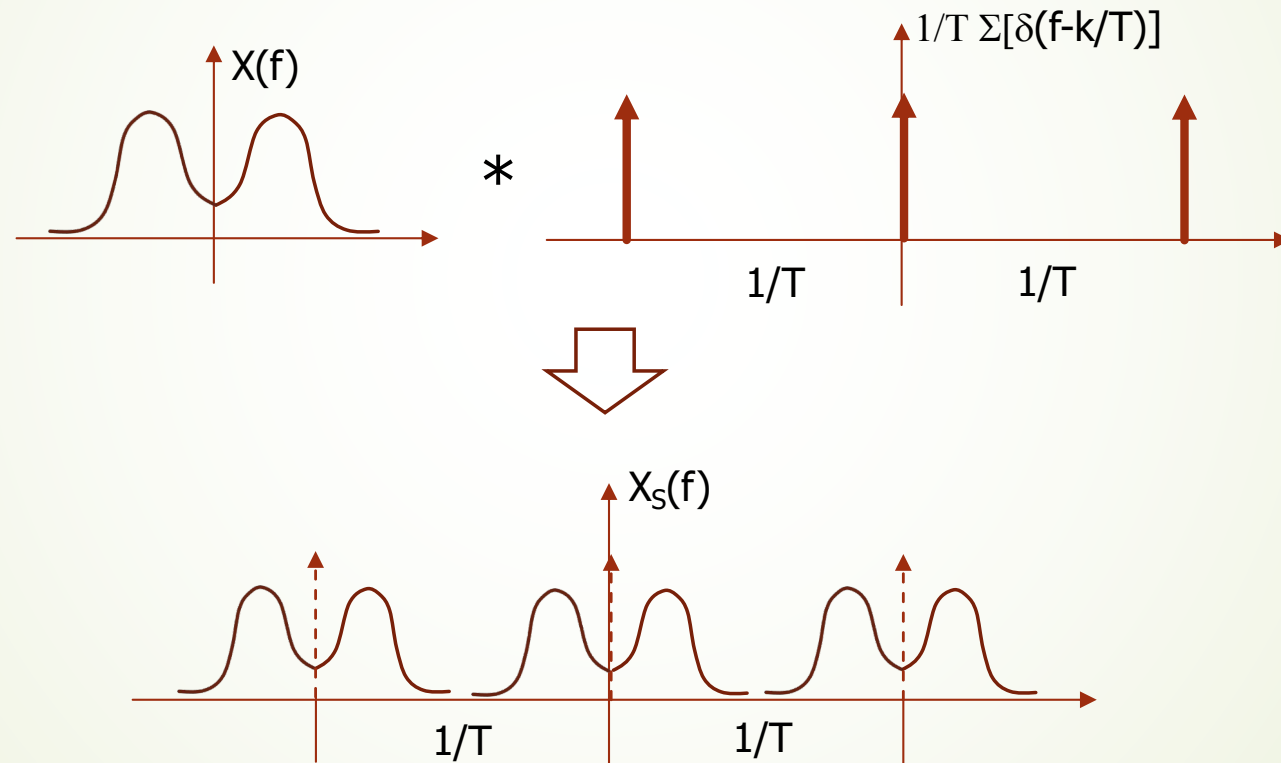
- Then, the frequency spectrum of the sampled signal $x_s(t)$ will be:

$$\mathcal{F}\left(x(t) \cdot \sum_{n=-\infty}^{+\infty} \delta(t - nT)\right) = \frac{1}{T} X(f) * \sum_{k=-\infty}^{+\infty} \delta\left(f - \frac{k}{T}\right)$$

NB. The convolution of a signal with an impulse simply translates the signal at the impulse location (easy to verify)

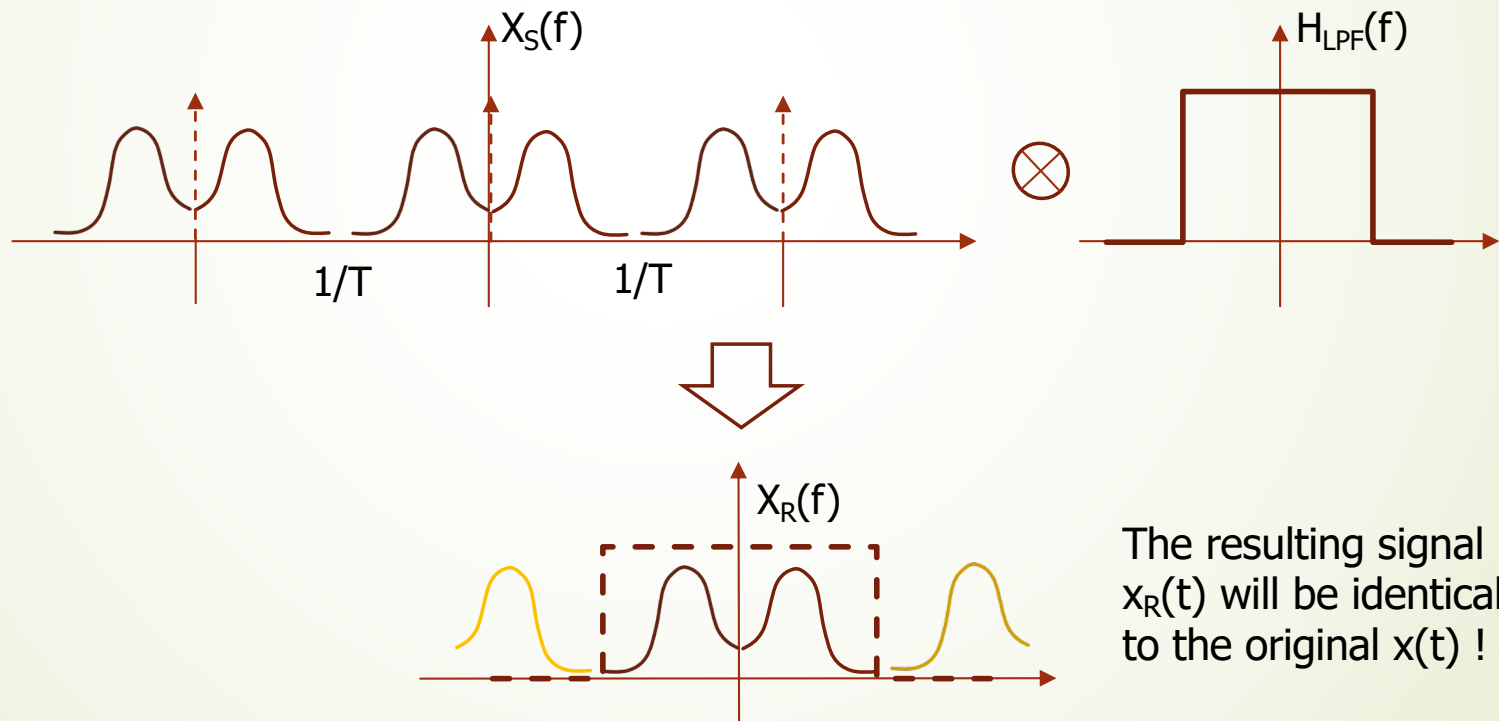
Sampling frequency

- Let us look at this result in a graphical form:



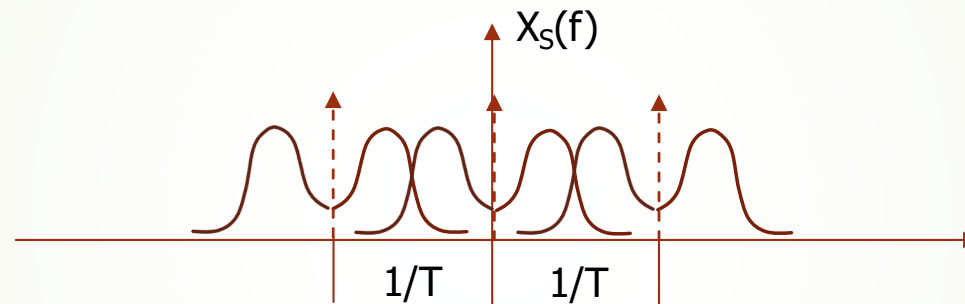
Recovering the continuous signal

- We can observe that $X_S(f)$ is a series of replicas of the spectrum of the original signal, placed at a distance $1/T$ one from each other
- If the situation is the one depicted in the previous slide, we can easily recover the original signal from the sampled one with a LPF

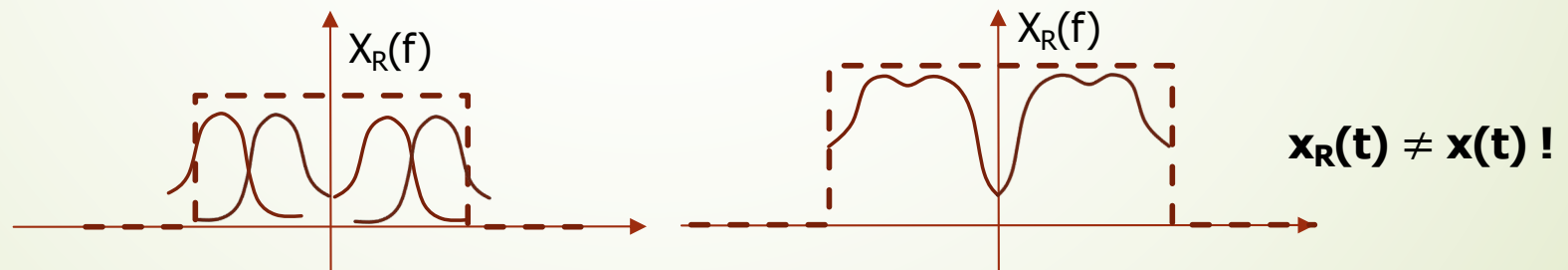


Aliasing

- In order this to happen, the sampling frequency should be high enough to prevent replicas to overlap each other



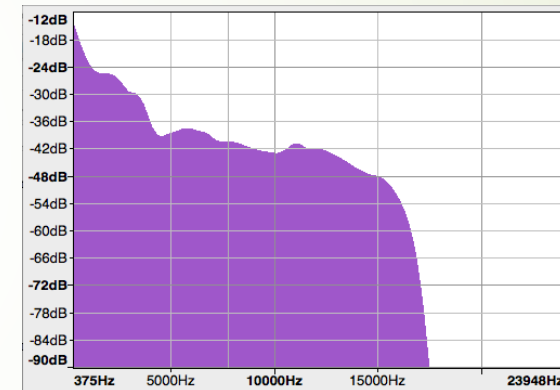
- If this happens the replicas will mix, making it impossible to reconstruct the original signal. This effect is called **aliasing**



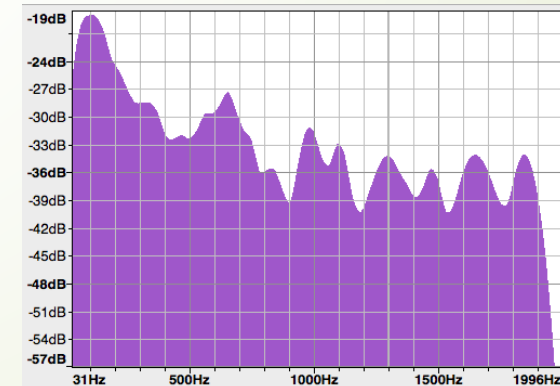
Example of aliasing

- Paradoxically, aliasing introduces unwanted high frequencies into the signal (lowpass components of the replica overlap to high frequency components of original spectrum)
- This reflects in spurious high frequencies in reconstructed signal

48k sample/sec

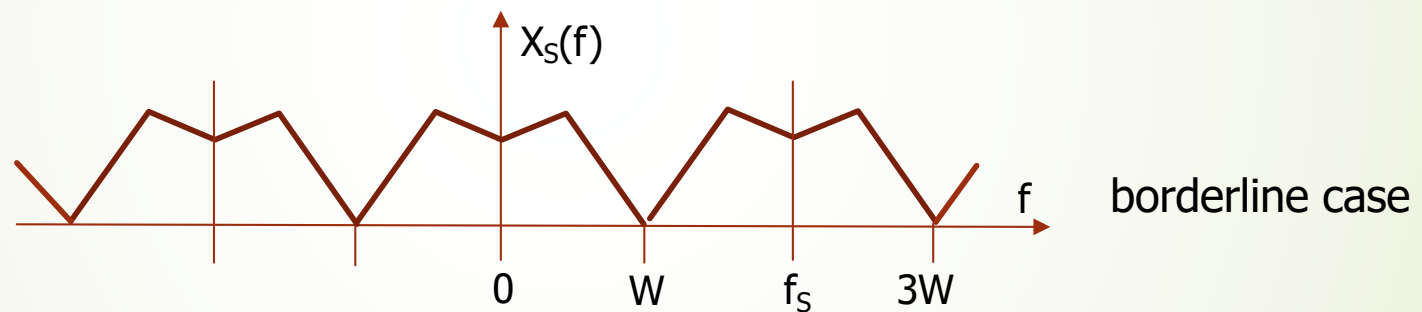


4k sample/sec



Avoiding aliasing

- To avoid aliasing, the sampling frequency must be set as to prevent overlapping of replicas
 - This means that the second replica must be placed at a distance $1/T$ equal or greater than twice the maximum frequency present in the signal (the so-called **signal bandwidth**)



$$f_s = \frac{1}{T} \geq 2W$$

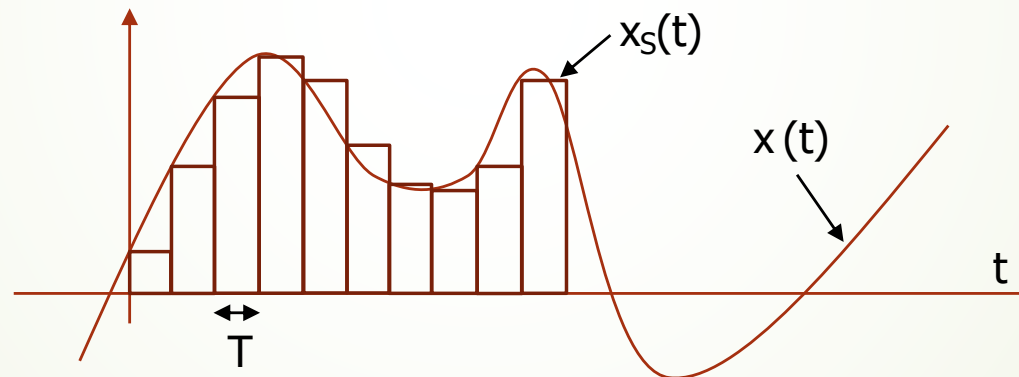
← Nyquist criterion

Nyquist criterion: example

- ▶ Human voice signals have a frequency spectrum ranging from 300Hz to 3500Hz approximately
- ▶ According to the Nyquist criterion so far introduced, we'll have to sample a human voice signal at a minimum of 7KHz
 - ▶ The sampler will produce 7000 samples per second
- ▶ Audio signals (e.g., music) require higher frequencies, in fact, we are able to hear frequencies up to 20KHz
 - ▶ Sampling frequencies for audio signals are 48kHz and up
 - ▶ Often, **oversampling** is applied (96KHz or more), to achieve higher sound resolution, better quality, and prevent aliasing

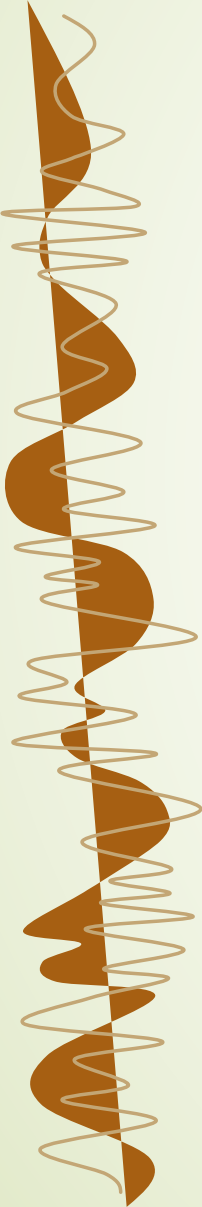
Real sampling

- The process we have described in the previous slides is **ideal**
- In the real world, it is impossible to generate impulses
 - The sampling signal will be some kind of rectangular function
 - The system takes the name of **sample&hold**, as it acts as a filter that gets the amplitude of the continuous signal at the sampling instant and retains it for the sampling period $T = 1/f_s$



- This results in a deformation of the signal spectrum that requires an **equalization** (we skip mathematical details)

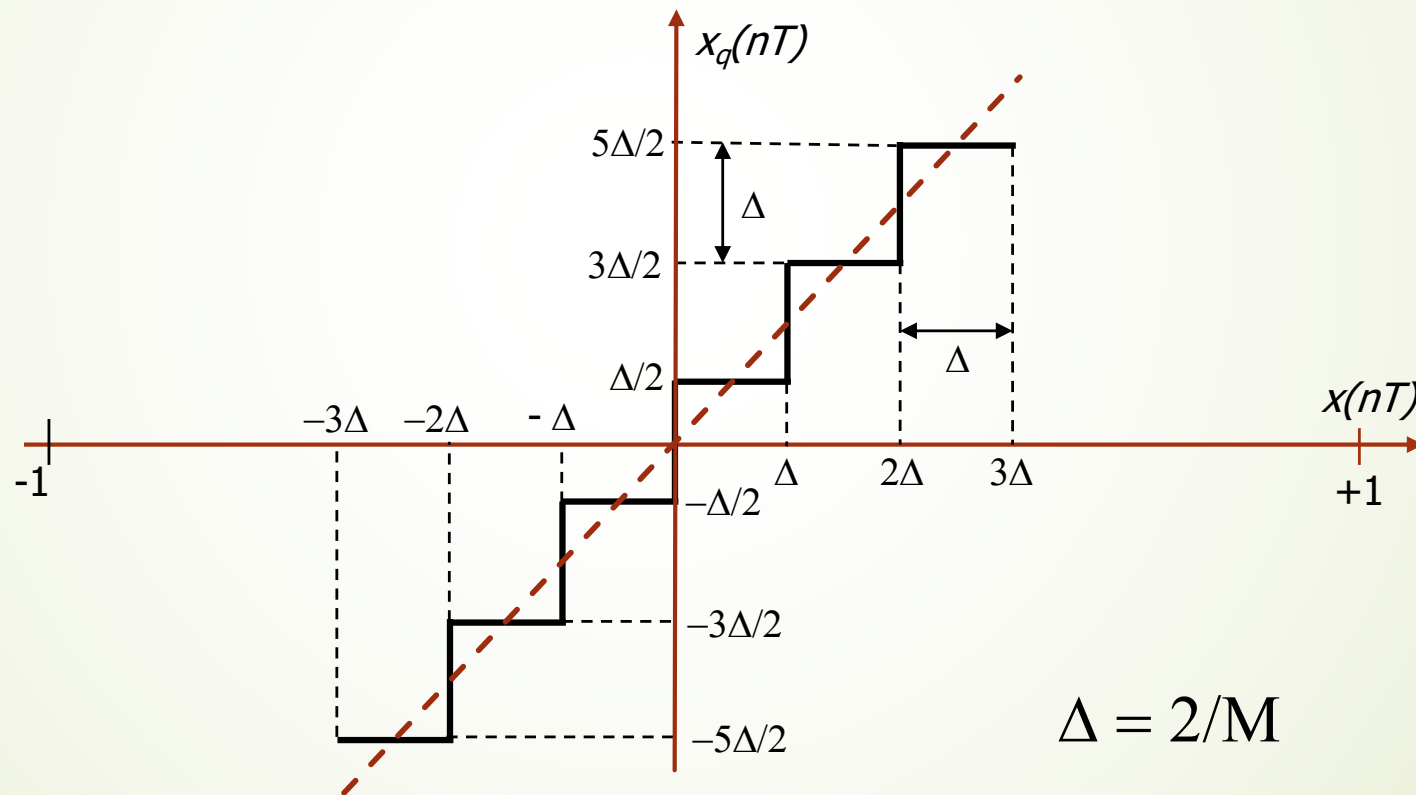
Quantization

- 
- In the sampled signal, each sample has still continuous amplitude values, and therefore could not be expressed in numerical way
 - It is necessary to discretize also the amplitude → **quantization**
 - A quantizer is a **non-linear** component that associates the continuous input samples to a pre-defined set of possible values, called **quantization levels**
 - The association between an input value (continue) and the corresponding (discrete) output value is performed according to a **minimum distance criterion**

*NB. This operation always introduces an error, called quantization error or **quantization noise** (distance between original and quantized samples)*

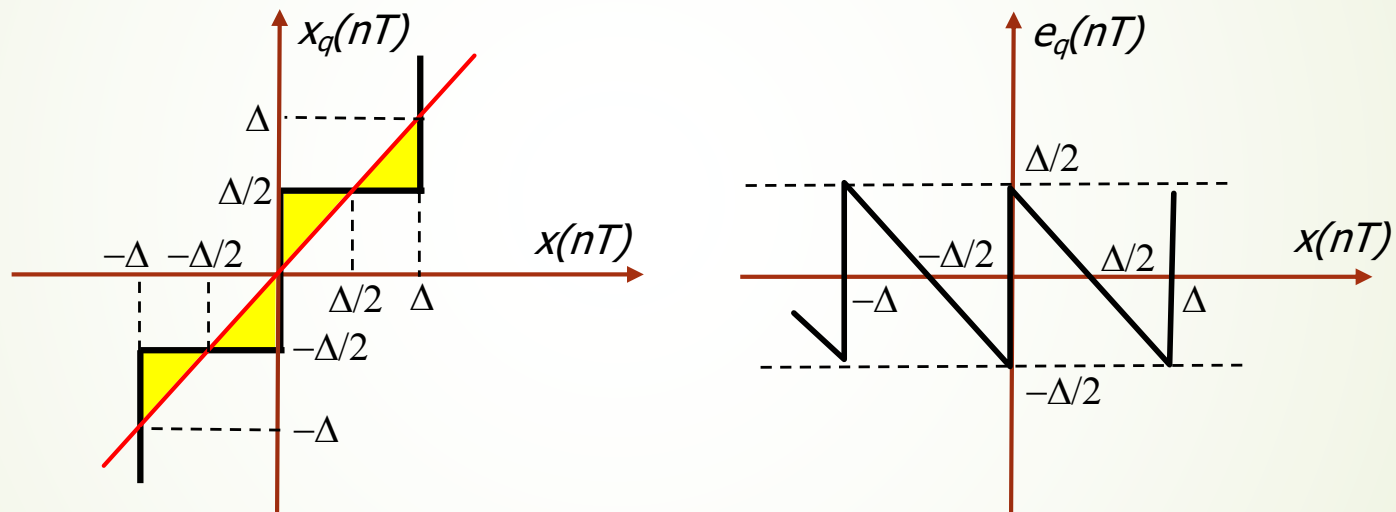
Uniform quantizer

- A uniform quantizer subdivides the input range in M equal ranges associated to M equally spaced levels
 - It could be described by a stair function with equal steps



Quantization error

- It is interesting to study the error introduced by the quantizer
- If we take a single step of the stair function, we can characterize the error function as a sawtooth wave

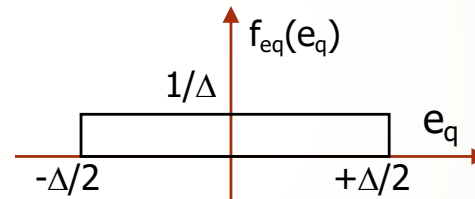


$$x(nT) = x_q(nT) + e_q(nT)$$

- e_q is called quantization error and behaves like a random noise

Quantization noise

- Although the quantization error is systematic, the input signal can be considered a random variable
- If we assume the input signal to be equally distributed, the error will inherit the same statistical properties, then:
 - e_q is a random variable equally distributed in $[-\Delta/2, \Delta/2]$



- We can calculate the power of the quantization noise as the mean square value of the error, $\overline{e_q^2}$
 - Since the expectation of e_q is zero, $\overline{e_q^2} = \sigma_q^2$, therefore:

$$\sigma_q^2 = \int_{-\Delta/2}^{\Delta/2} e_q^2 f_{e_q}(e_q) de_q = \frac{1}{\Delta} \int_{-\Delta/2}^{\Delta/2} e_q^2 de_q = \frac{1}{\Delta} \frac{e_q^3}{3} \bigg|_{-\Delta/2}^{\Delta/2} = \frac{\Delta^2}{12}$$

Signal to noise ratio

- If we compare the power of the noise with the power of the signal, we have an estimate of how good the noisy signal is
 - This measure is called SNR, and is typically measured in logarithmic scale (dB, or decibel)
- The power of the signal can be calculated the same way we did for the noise (it is a random variable equally distributed in $[-1,1]$)

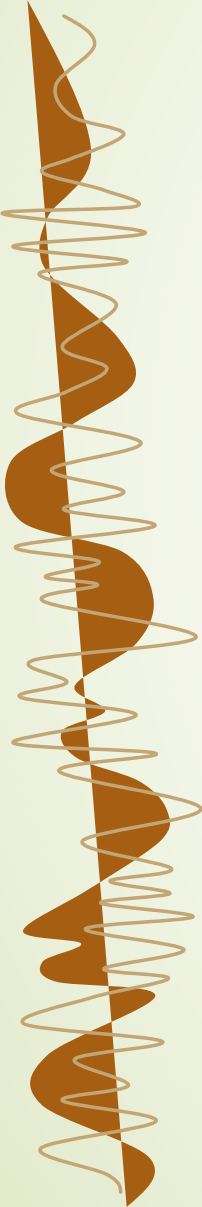
$$\sigma_x^2 = \int_{-1}^1 x^2 f_x(x) dx = \frac{1}{2} \int_{-1}^1 x^2 dx = \frac{1}{2} \frac{x^3}{3} \Big|_{-1}^1 = \frac{4}{12}$$

then: $SNR = 10 \log_{10} \left(\frac{\sigma_x^2}{\sigma_q^2} \right) = 10 \log_{10} \left(\frac{\frac{4}{12}}{\frac{\Delta^2}{12}} \right) = 10 \log_{10} \left(\frac{4}{\Delta^2} \right)$

but: $\Delta = \frac{2}{M} = \frac{2}{2^B}$ then $SNR = 10 \log_{10}(2^{2B}) \approx 6B$

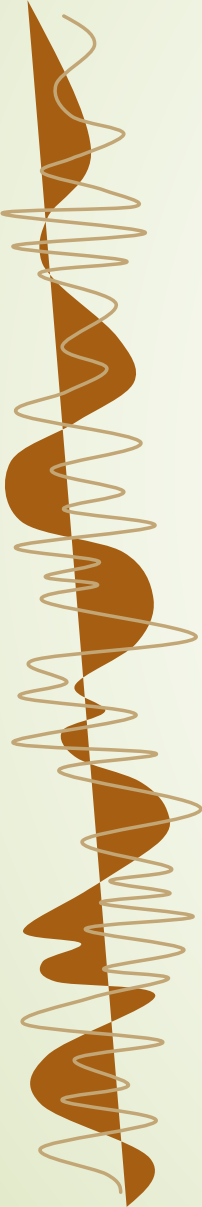
- **Every bit added to the quantizer increases the SNR of 6dBs !**

SNR practical examples

- 
- SNRs below 20dBs are typically associated to poor quality signals
 - Increasing the dBs provides better and better signals, e.g.:
 - Voice telephone systems → 40dB
 - AM radio → 50dB
 - FM radio → 70 dB
 - CD player → 90 dB
 - HiFi studio recording → 120 dB
 - To achieve 90dB quality in music CDs, we have to quantize at 16 bits/sample ($16 \times 6 = 96$), i.e., 65536 levels

NB. Multiples of 8 bit/sample are usually preferred for practical reasons (byte alignment in computers)

Encoding

- 
- The last step of the analog-to-digital conversion is the encoding of the quantized samples
 - Since at this point we have a finite number of discrete levels, we can code them with a finite number of bits
 - Given M quantization levels, we will need $\log_2 M$ bits, or the larger nearest integer if M is not a power of 2
 - Typically we will choose M as a power of 2: $M = 2^B$
 - The continuous signal $x(t)$ has been finally converted to a discrete sequence $x(n)$, where each element of the sequence is represented by a B -bit binary code

We have achieved a binary representation of our signal !

Bitrate

- Once the signal is sampled, quantized and coded, it generates a stream of bits
- For time signals it is useful to calculate the number of bits generated in a time unit (1 sec): this value is called **bitrate**
- A signal of bandwidth W , sampled at the Nyquist frequency and quantized as to achieve around Q dB SNR, will generate a stream at bitrate:

$$bitrate = 2W \left\lceil \frac{Q}{6} \right\rceil$$

- For instance, an audio signal with bandwidth 20KHz and desired quality around 50dB will generate a bitrate:

$$2 \times 20 \times 10^3 \times 8 = 320 \text{ Kbps}$$

Numerical systems

- To deal with numerical signals, we need **numerical systems**
- A numerical system is a system that accepts numerical signals in input and produces numerical signals in output
- In principle, we can convert an analog system into a numerical one by:
 - sampling the impulse response
 - in the Fourier domain, applying the discrete transform
- In the first case, we typically obtain an infinite number of samples in the impulse response (**IIR**)
 - It is possible to cut the response with a windowing (→ aliasing)
 - What we obtain is the so-called **FIR (finite impulse response)** filter

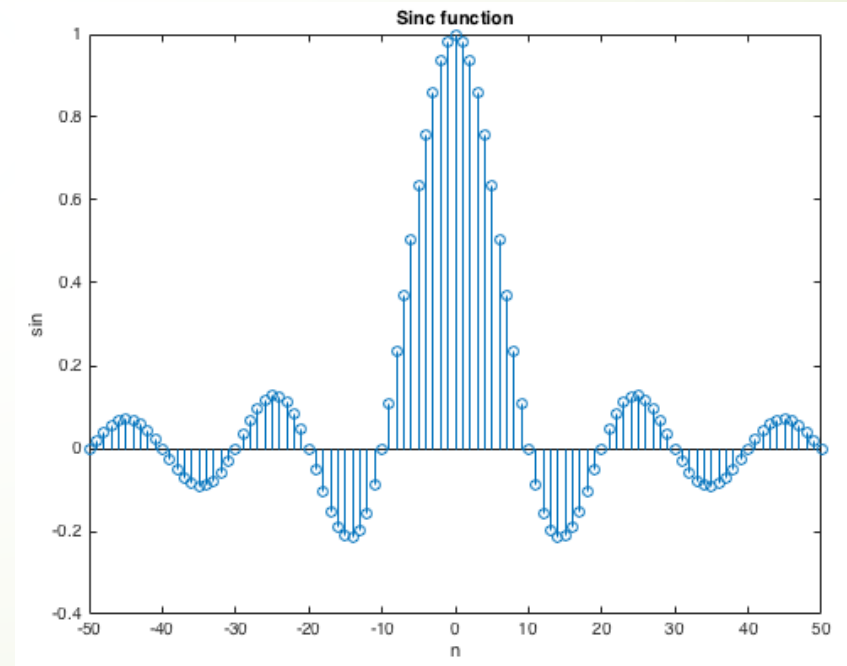
NB. A thorough discussion requires complex math (z-transform)

Example: LPF

- Ideal low-pass has a rectangular frequency response function $H_{\text{LPF}}(f)$
- The corresponding impulse response is a sinc function
 - When sampling the sinc at Nyquist rate we get:

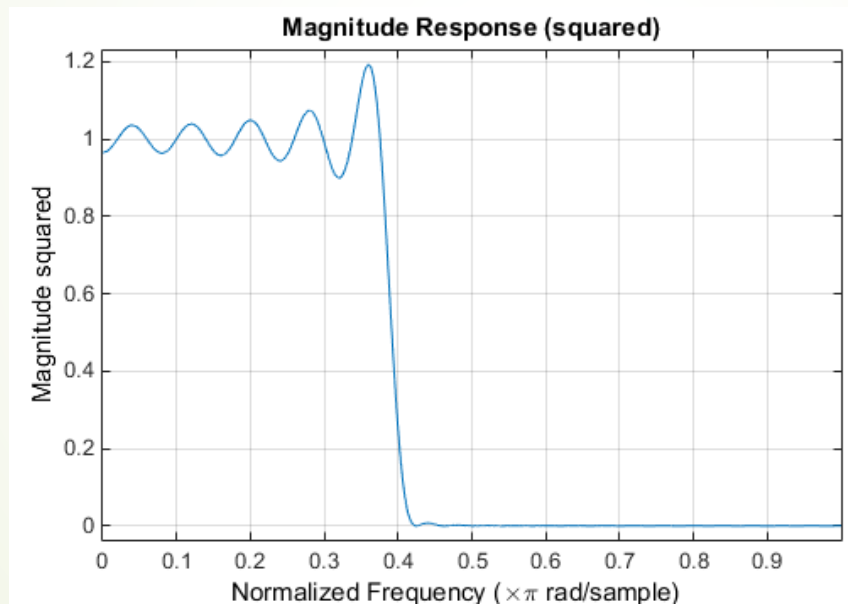
$$h_D(n) = 2f_s \frac{\sin 2\pi n f_c}{2\pi n f_c}$$

AN INFINITE NUMBER OF SAMPLES!



Example: LPF

- The above filter cannot be implemented in practice
 - Infinite number of operations per input sample
- We could simply clip the sinc at some point 'M' large enough
 - The windowed response $h_{DW}(n)$ introduces an **error**:



Frequency response shows:

- A transition region
- Some ripples

NB. More sophisticated techniques may limit these problems

System response in numerical domain

- Once we have calculated the discrete impulse response $h_{DW}(n)$, we can calculate the output through a **discrete convolution**

- It is the discrete version of the analog convolution:

$$y(n) = \sum_{k=-\infty}^{\infty} x(k)h_{DW}(n-k)$$

- If the number of samples in the filter is limited to M , we'll have:

$$y(n) = \sum_{k=-M/2}^{M/2} x(k)h_{DW}(n-k)$$

- M product-sum operations per sample!

Discrete convolution: example

$$h_{\text{DW}}(n) \begin{bmatrix} 1 & 0 & 2 \end{bmatrix} \rightarrow h_{\text{DW}}(-n) \begin{bmatrix} 2 & 0 & 1 \end{bmatrix}$$

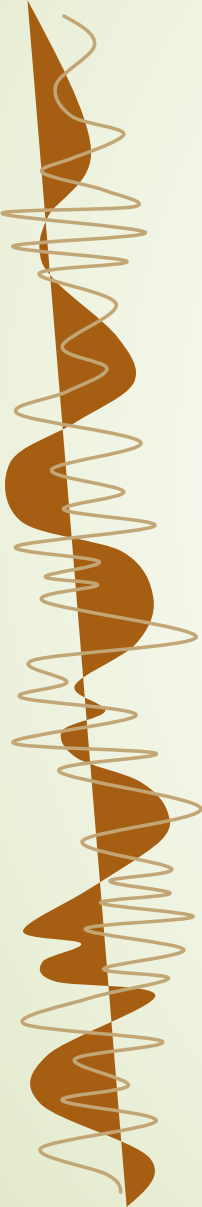
$$x(n) \begin{bmatrix} 1 & 2 & 3 & 1 & 0 & 2 \end{bmatrix}$$

$$\begin{array}{ccccccccccccccc} & & & & 1 & 2 & 3 & 1 & 0 & 2 & & & & \\ 2 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & \end{array}$$

$$y(n) \begin{bmatrix} 0 & 1 & 2 & 5 & 5 & 6 & 4 & 0 & 4 & 0 \end{bmatrix}$$

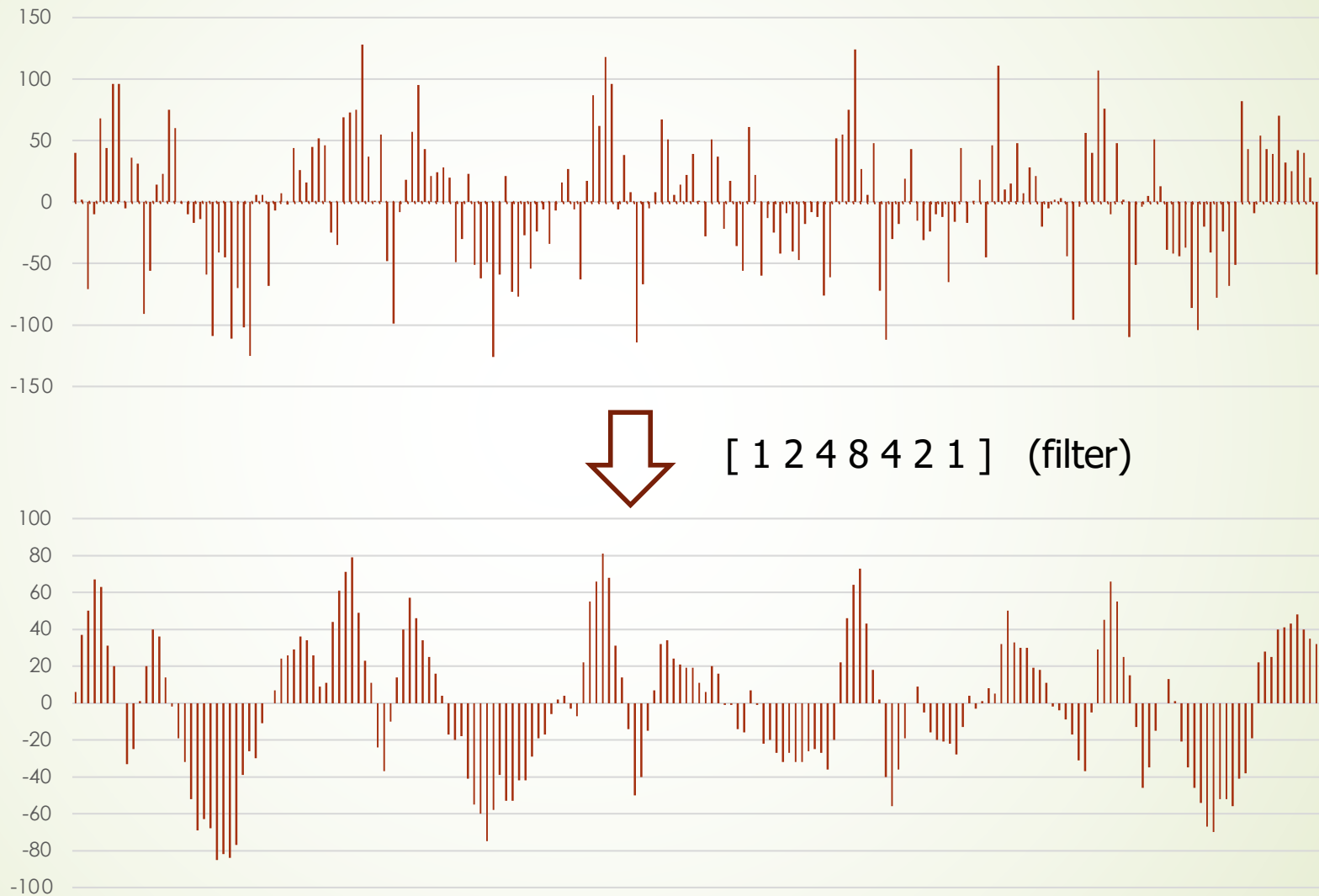
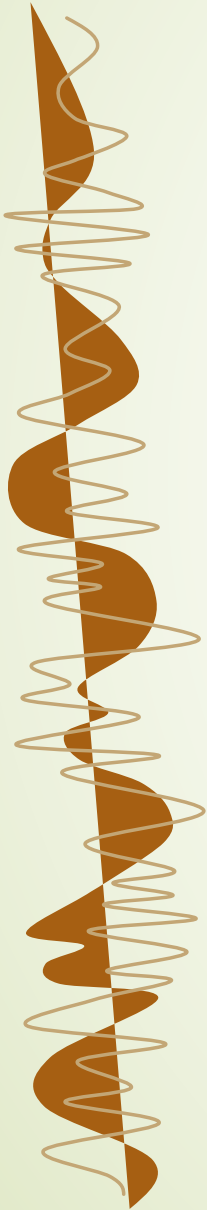
no overlap $0 \times 1 + 2 \times 2 + 3 \times 3 + 1 \times 1 + 0 \times 0 + 2 \times 2$

Discrete convolution: pseudo-code



```
int input[N], output[N]           // input and output vectors
int imp_resp[M]                   // impulse response
int hl = floor(M/2)               // half length of filter
load(input), load(imp_resp)       // load input data and kernel
for i in hl ... N-hl-1 {          // scan input vector w/o borders
    tmp = 0                       // init var
    for j in -hl ... hl {         // scan filter
        tmp += (input[i+j] * imp_resp[j+hl]) // filter
    }
    output[i] = tmp               // write result to output vector
}
```


Example: LPF of a noisy wave



Frequencies in numerical domain

- ▶ The Fourier frequency representation has a discrete counterpart, called **Discrete Fourier Transform** (DFT)
 - ▶ Given a finite sequence of samples, the DFT converts it into a finite sequence of frequency-domain coefficients of the same length
 - ▶ Each DFT coefficient is associated to a complex sinusoid

Given a discrete input signal $\{x_n\}$ of length N :

$$\{x_n\} = x_0, x_1, \dots, x_N$$

Its DFT is the discrete sequence $\{X_k\}$ of length N given by:

$$X_k = \sum_{n=0}^{N-1} x_n \cdot e^{-j\frac{2\pi}{N}kn}$$

DFT

DFT basis functions

- ▶ The exponential terms represent complex sinusoids (Eulero) at multiples 'k' of the fundamental frequency

$k=0 \rightarrow$ frequency zero (DC-term)

$k=1 \rightarrow$ fundamental frequency (1 cycle over N samples)

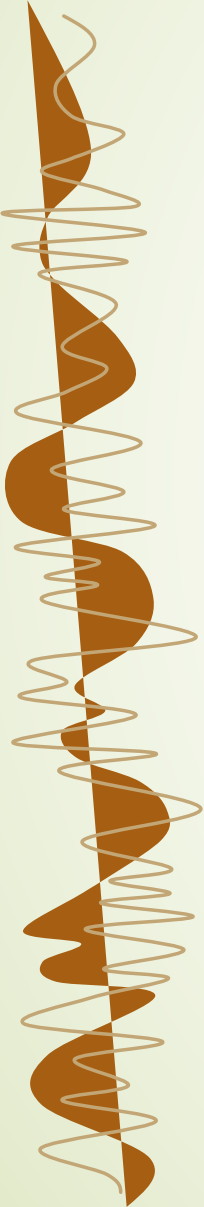
...

$k=K \rightarrow$ K-th harmonic frequency (k cycles over N samples)

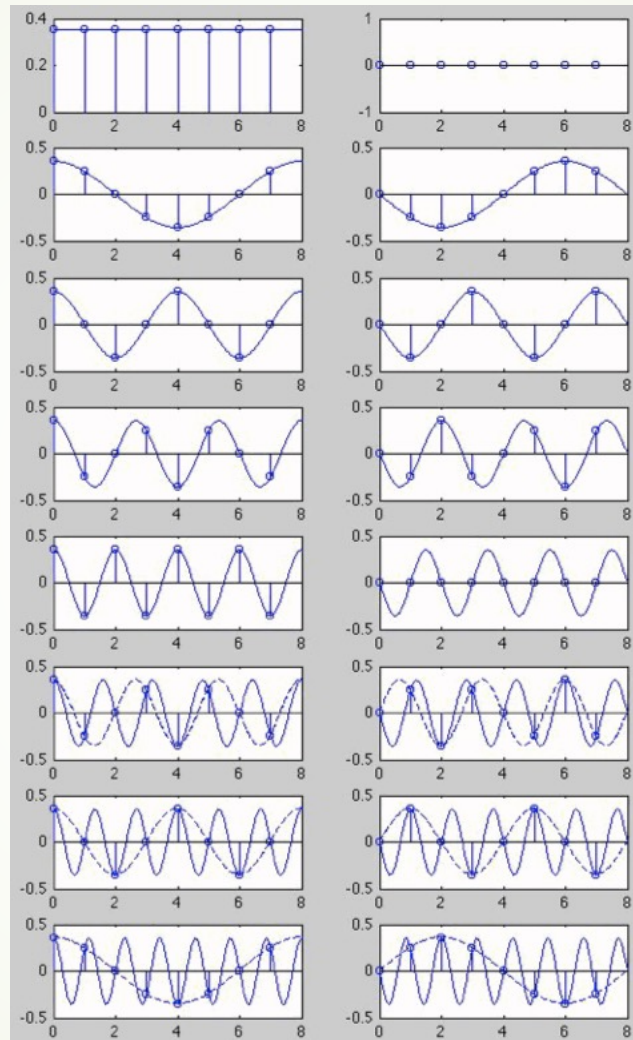
...

$k=N-1 \rightarrow$ highest frequency (N-1 cycles over N samples)

- ▶ Each sinusoid is sampled in N equally spaced points (aliasing occurs at higher harmonics)



DFT basis functions (graphical)



cosine waves

sine waves

← Continuous wave

← Fundamental frequency

← Highest frequency that can be represented with 8 samples

Inverse DFT (IDFT) and Fast transform

- Given the DFT coefficients, it is possible to reconstruct the original sequence by calculating the weighted sum of the basis functions

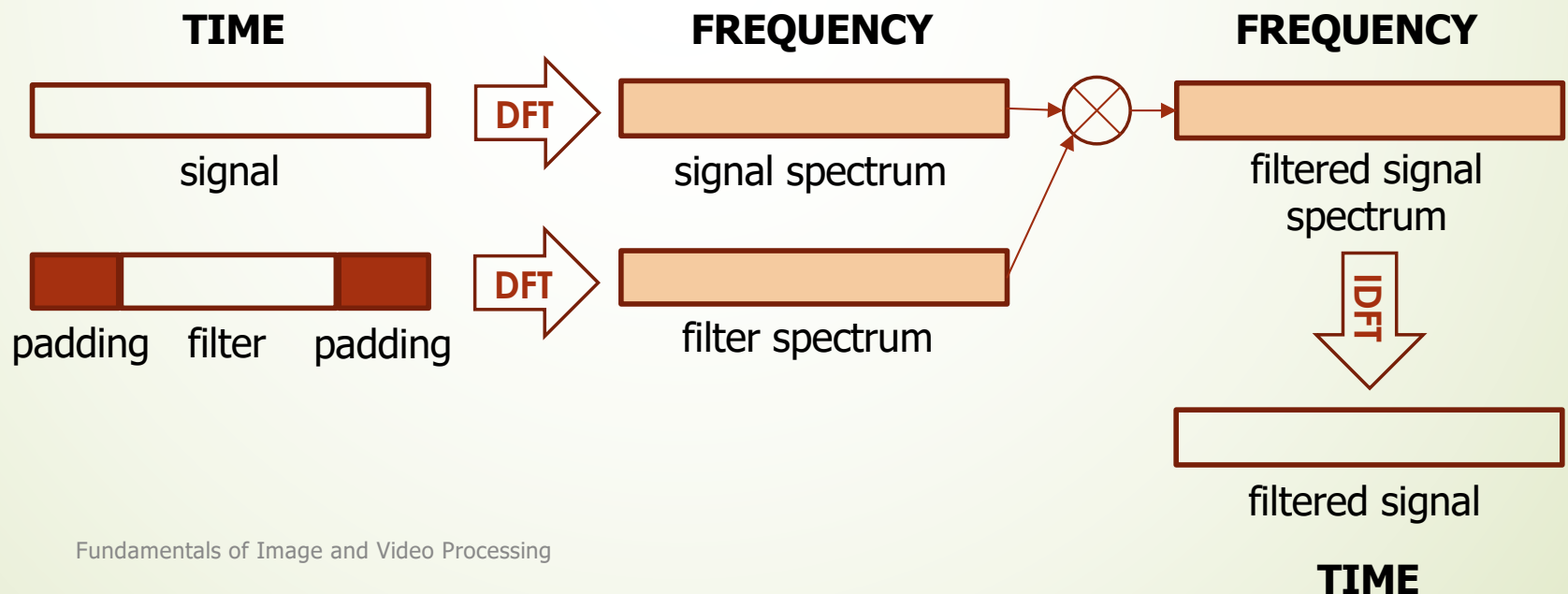
$$x_n = \frac{1}{N} \sum_{k=0}^{N-1} X_k \cdot e^{j \frac{2\pi}{N} kn}$$

IDFT

- In general, direct and inverse DFT require N product-sum operations per sample, with an overall **complexity $O(N^2)$**
- There exists a fast algorithm called Fast Fourier Transform **FFT** that provides the same result with **$O(N \log N)$ complexity**
 - FFT is largely diffused and implemented in most signal processing and math software libraries

Discrete filters in DFT domain

- Exactly the same way we can work in the Fourier domain with continuous signals, we can do it with discrete signals
 - Discrete time **convolution becomes a product** in DFT domain
 - In the discrete domain we just have to multiply numerical samples
 - In order to do that, we must ensure that sequences have the same length (**zero-padding** is applied to the shortest sequence)



What we've learned in this section

- Digital signals are more robust and easier to manipulate
- Analog to digital conversion requires sampling (discretization in time), and quantization (discretization in amplitude)
- Sampling does not lose information, if it is performed correctly (sampling theorem), otherwise it produces artifacts (aliasing)
- Quantization always loses information, but such loss can be limited to an acceptable level by using enough bits
- To handle numerical signals we need numerical systems: as for the analog case, for LTI systems we can define a numerical impulse response, and a numerical convolution
- An equivalent of the frequency representation also exists for digital signals (DFT), with similar properties and use of its analog counterpart