

Intelligent Signal Processing

Audio Digitization

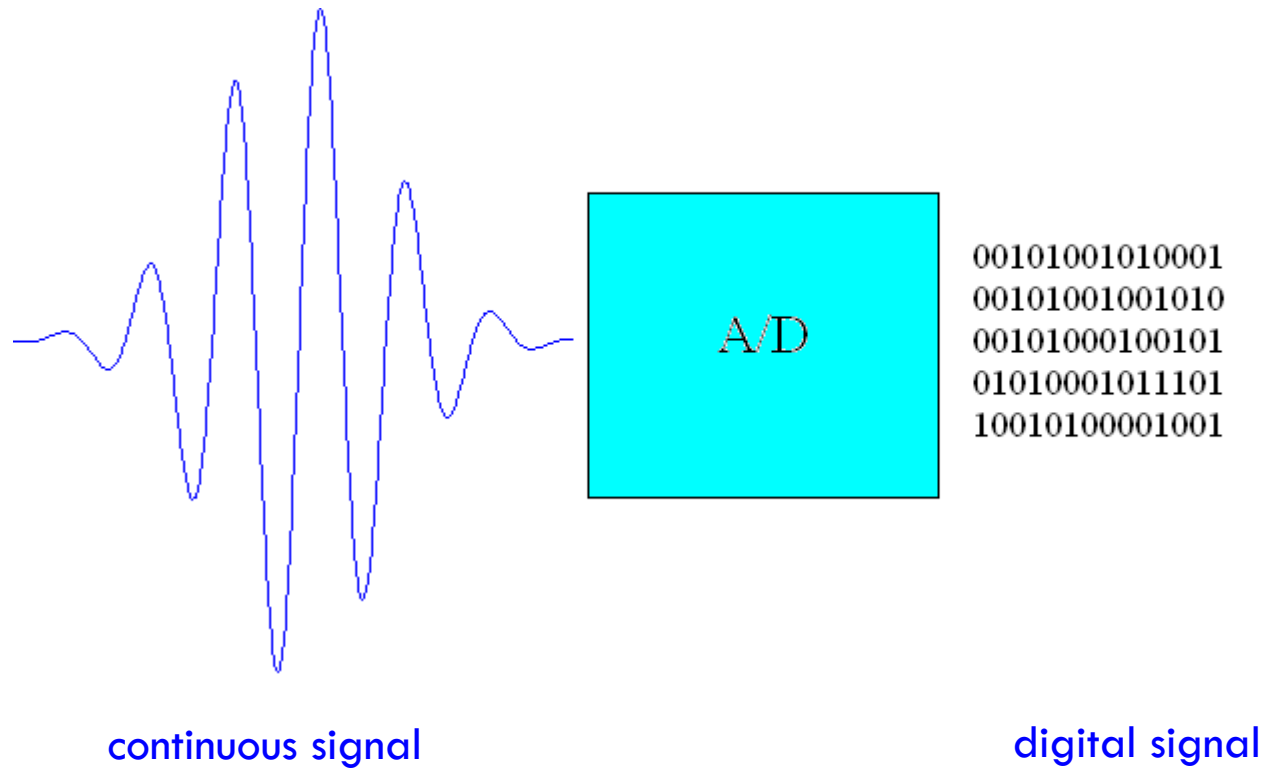
Angelo Ciaramella

Introduction

- The signals of **everyday reality** are **continuous** in time and in amplitude
 - voices
 - urban noise
 - musical listening
 - ...
- In modern acquiring and transmission systems, **information is binary** (0/1)



Sound digitization

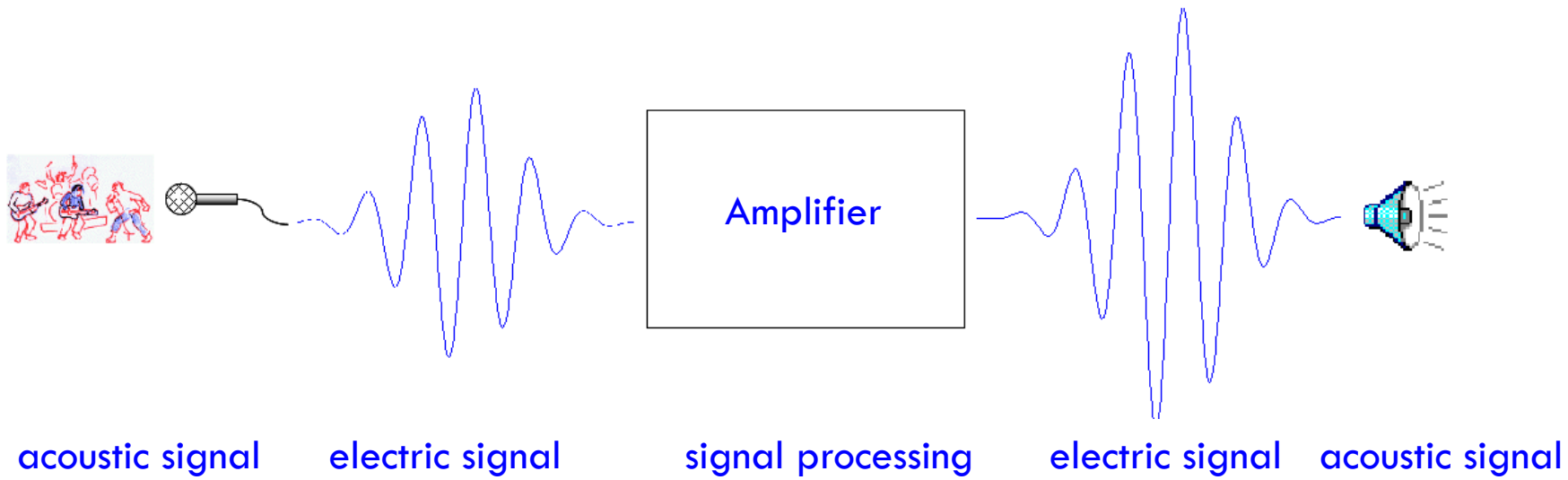


Transduction

- **Transduction** is the mechanism for transforming a physical signal into an electric and viceversa
 - **Sensors**
 - transduction from acoustic to electric signals
 - e.g., microphone
 - **Actuators**
 - transduction from electric to acoustic signals
 - e.g., speakers



Transduction scheme



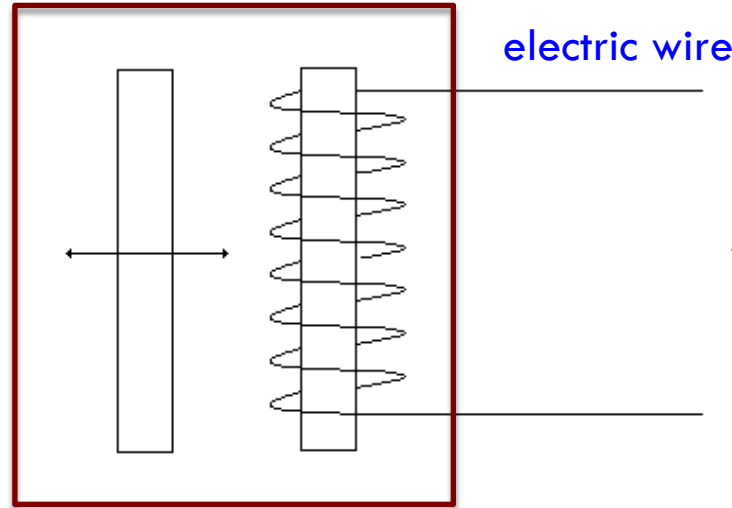
Principle of electromagnetic induction



acoustic sound



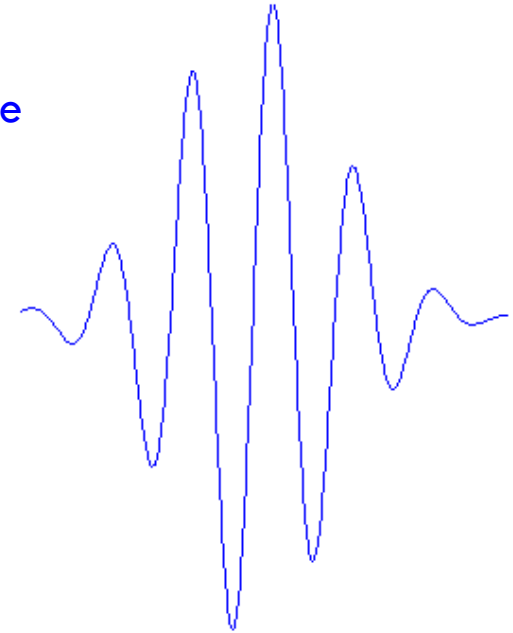
microphone



moving magnet

fixed magnet

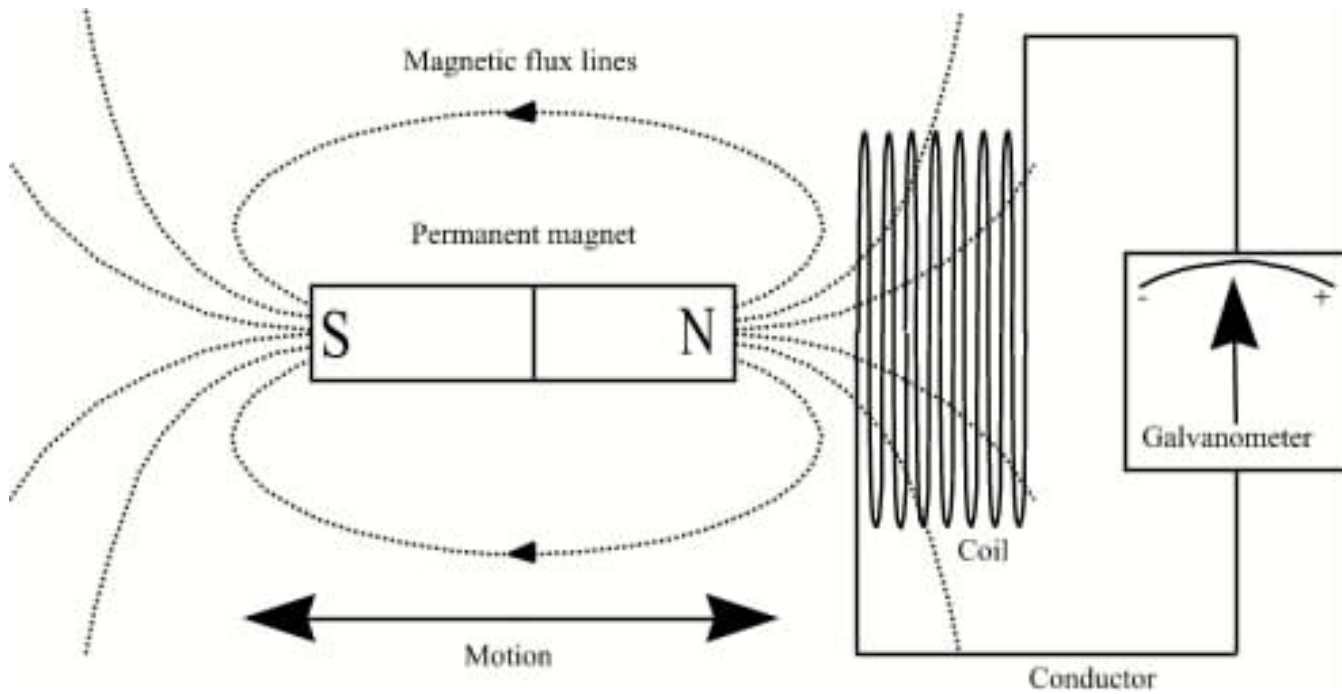
electric wire



electric sound



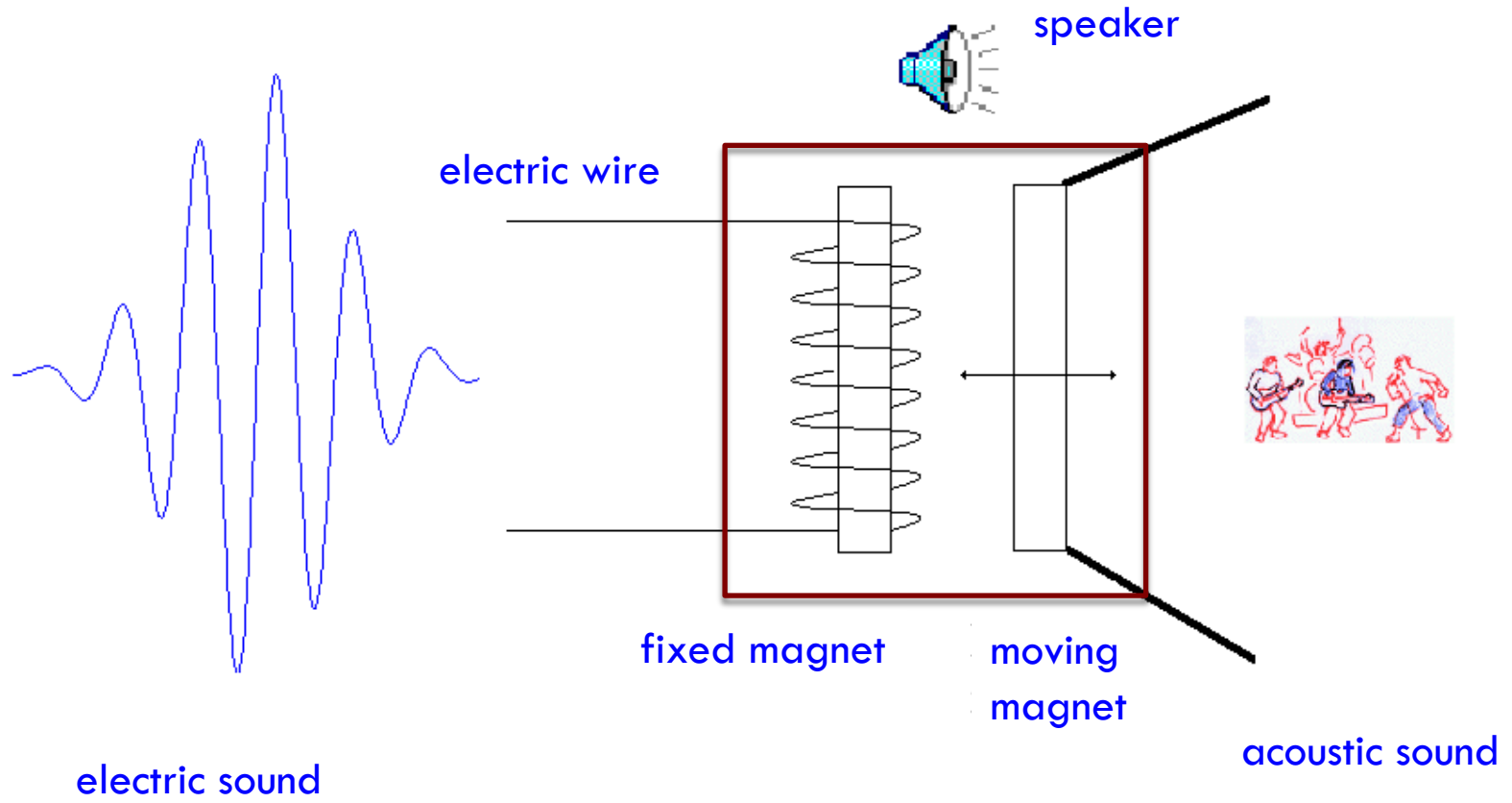
Faradays Law



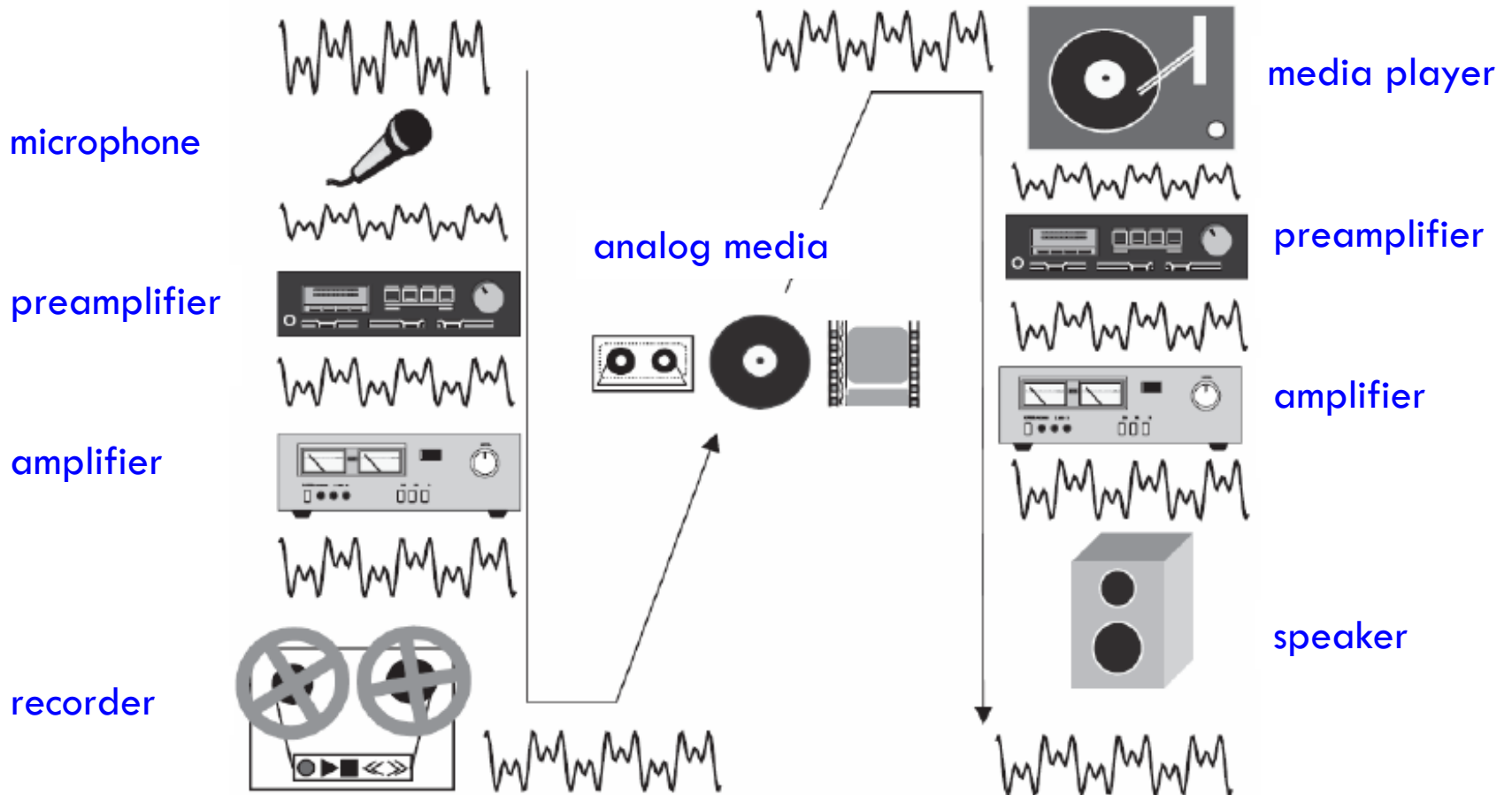
fixed magnet



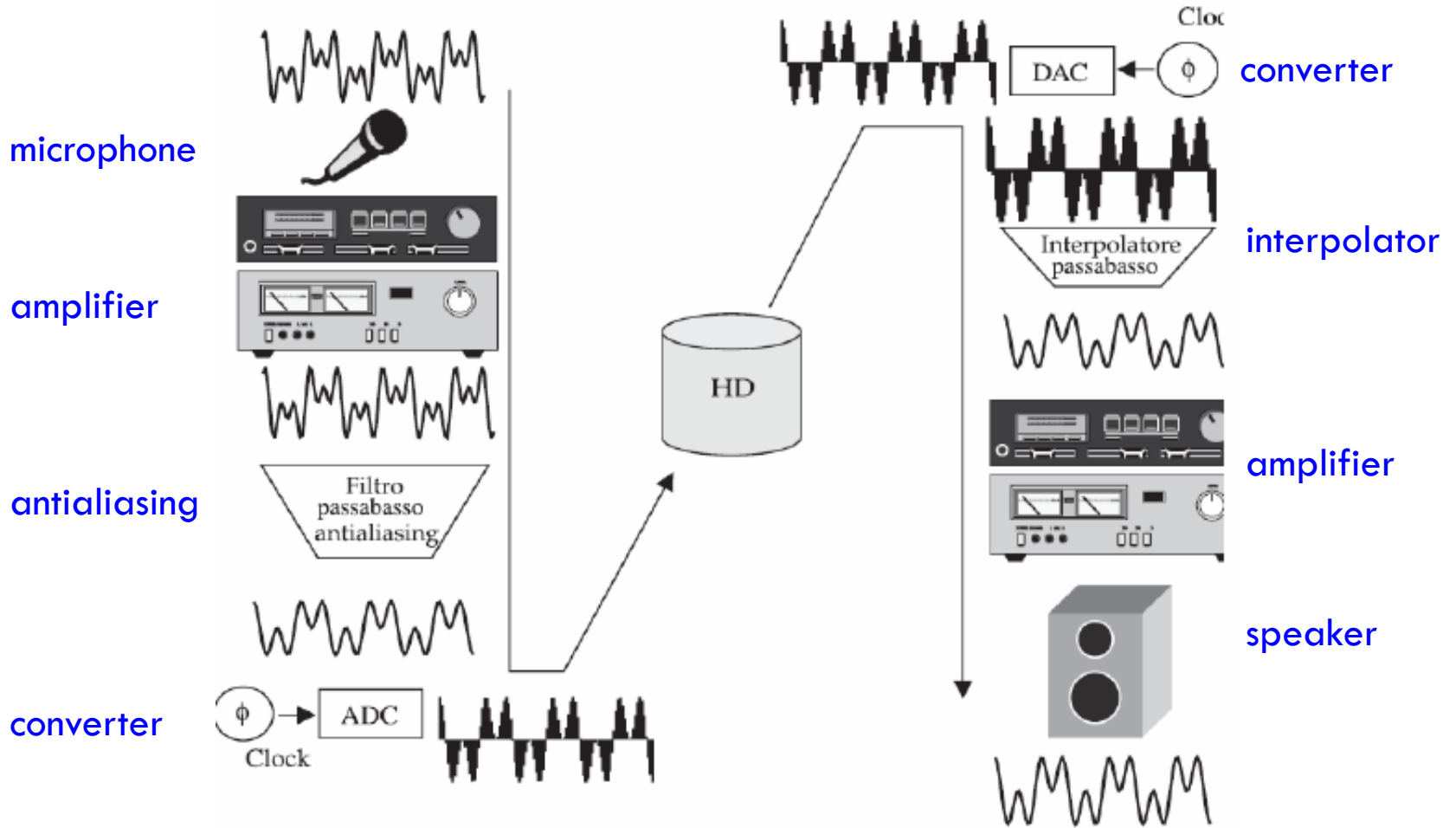
Electromagnetic induction



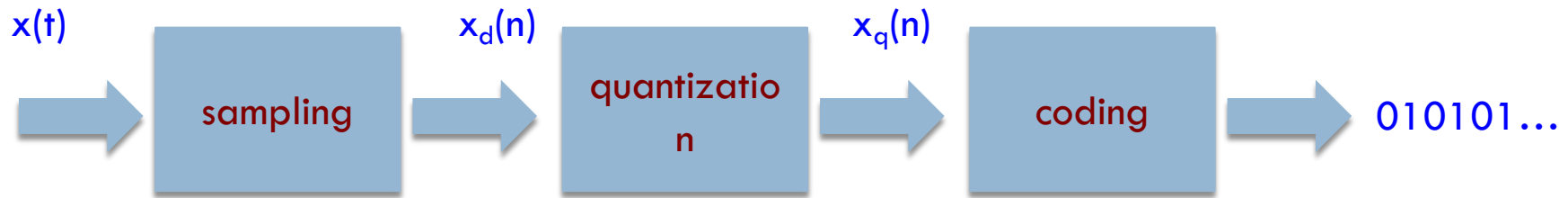
Analog components



Digital components



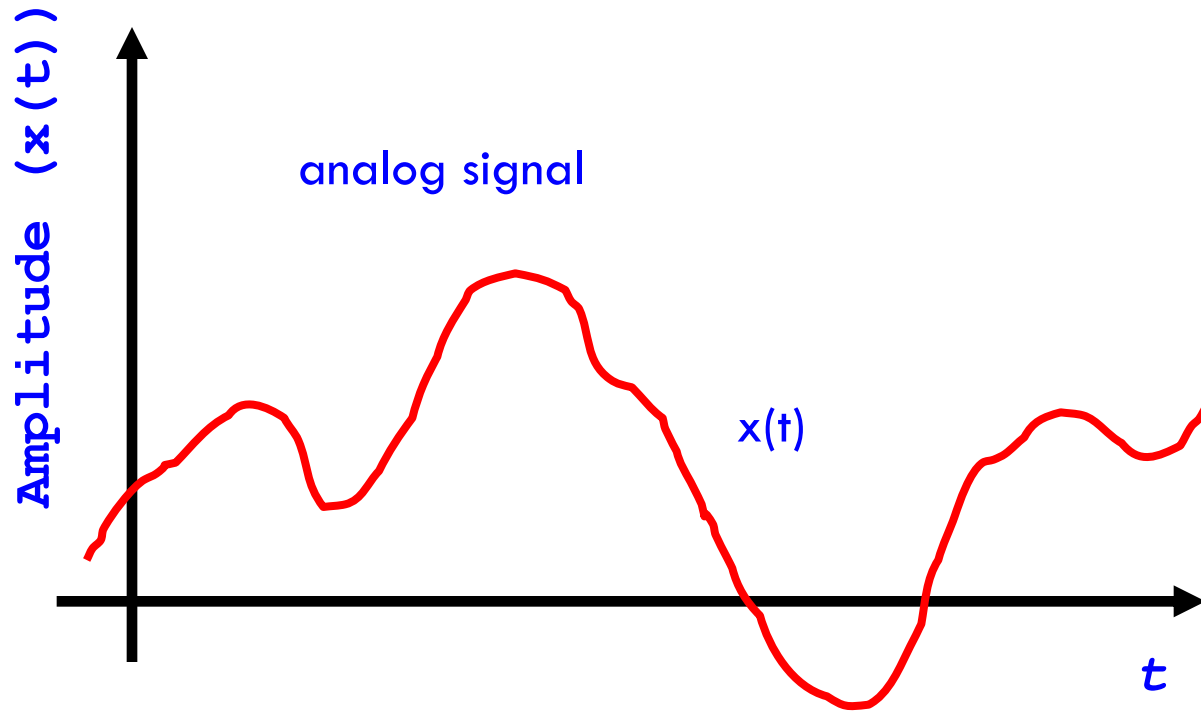
Analog to Digital Converter



Phases of the ADC process



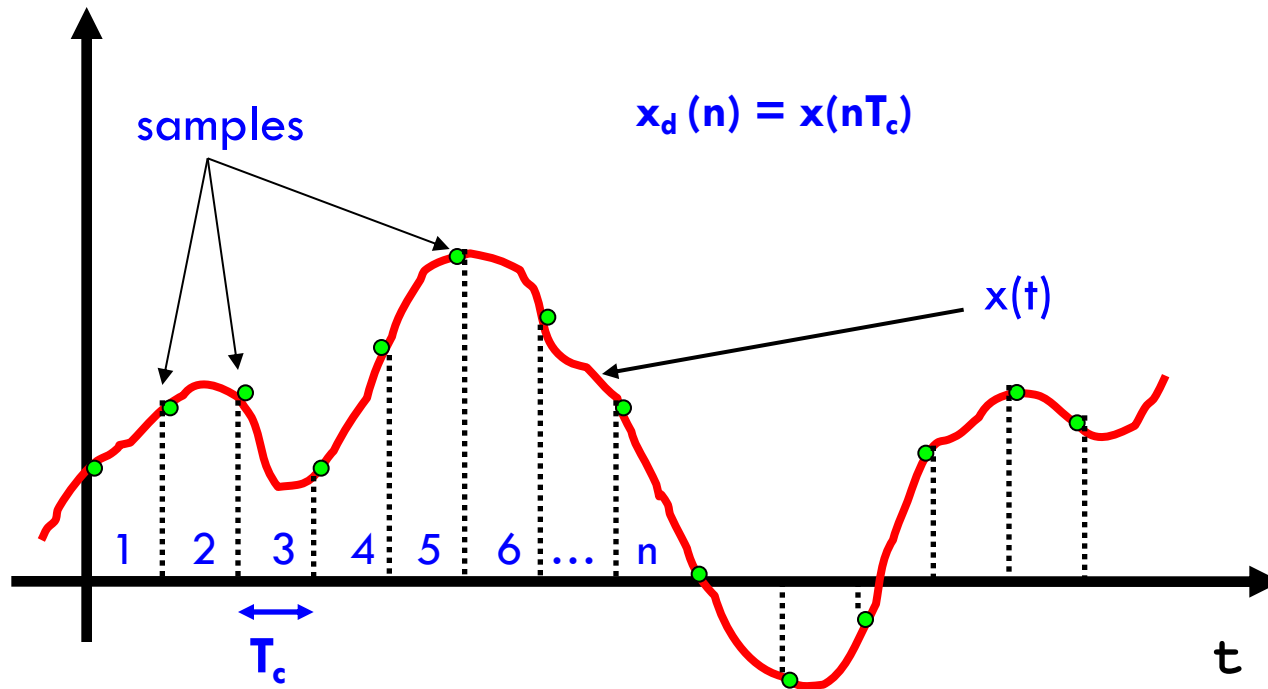
Analog signal



Analog signal temporal representation



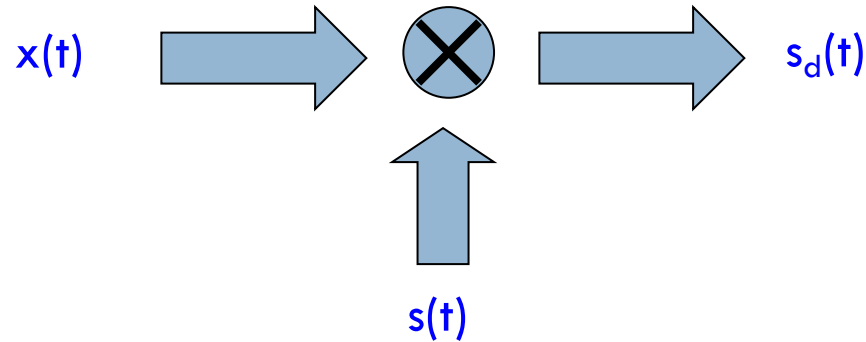
Analog signal sampling



Analog signal temporal representation



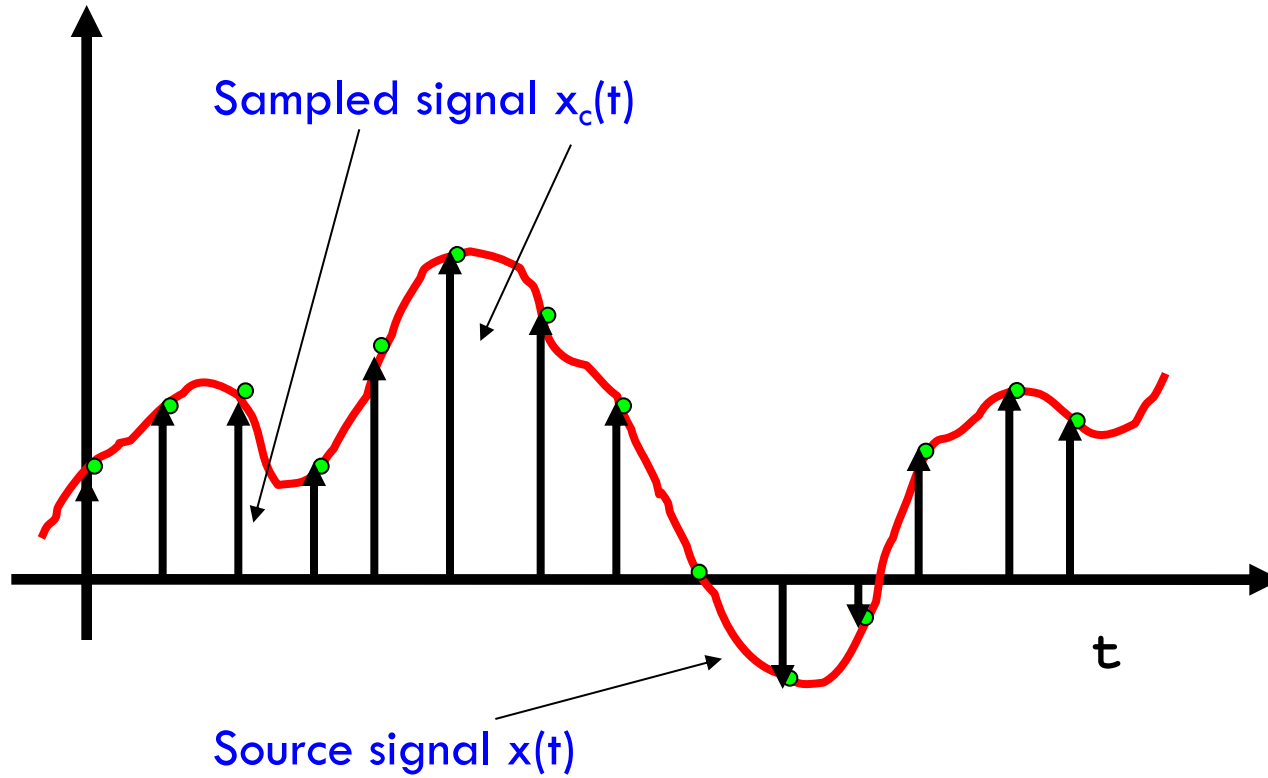
Pulse Code Modulation (PCM)



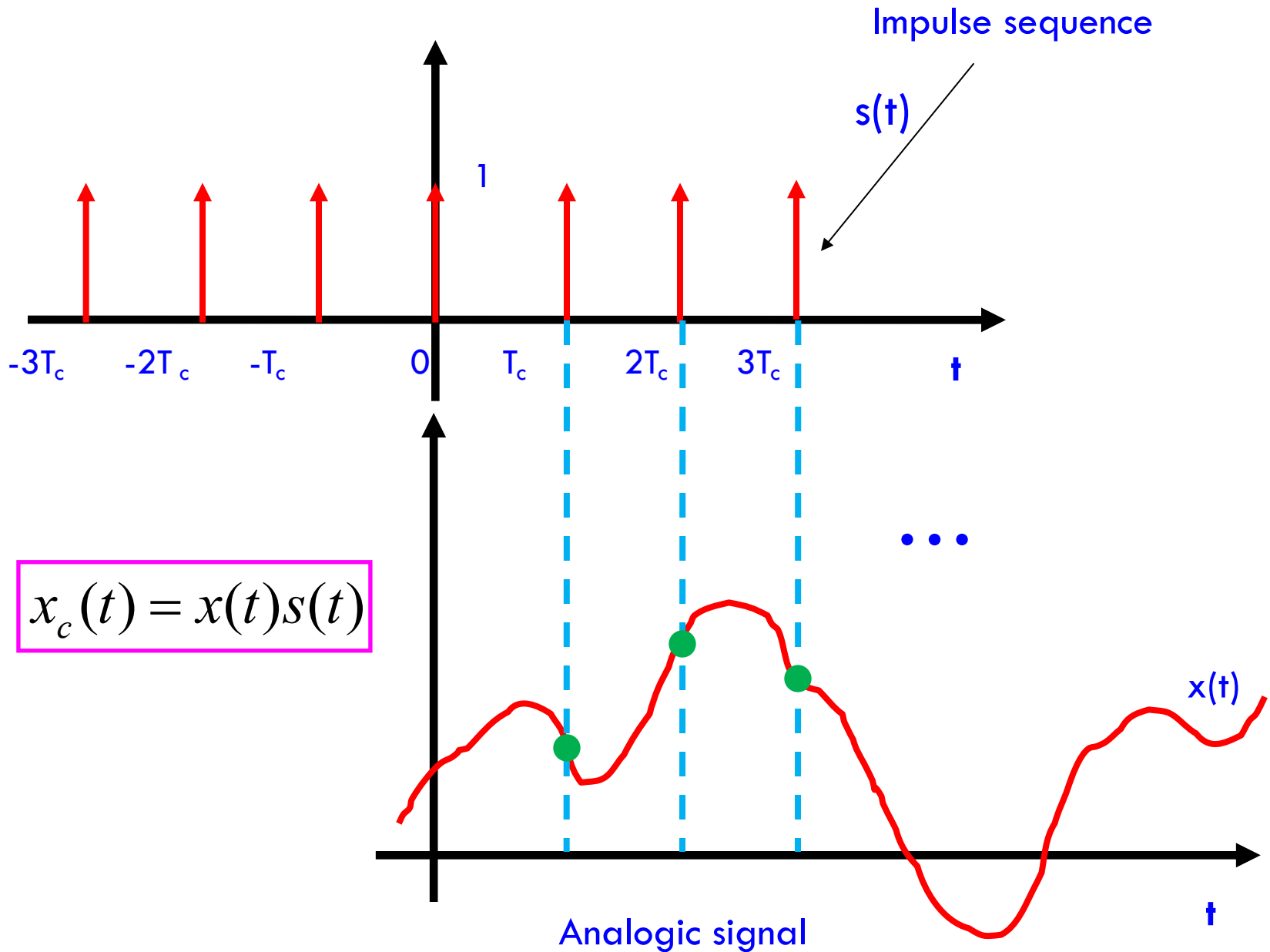
$s(t)$ is a periodic impulse sequence ($\delta(t)$, delta di Dirac)



Pulse Code Modulation (PCM)



Pulse Code Modulation (PCM)



Sampling theorem

- The **minimum sampling frequency** which is necessary to avoid distortions in the signal **reconstruction**
- Introduced by Harold **Nyquist**, and appeared in 1949 in an article authored by **E. C. Shannon**
- Given a signal with a limited and known bandwidth, the minimum sampling frequency of this signal must be **at least twice** its highest frequency



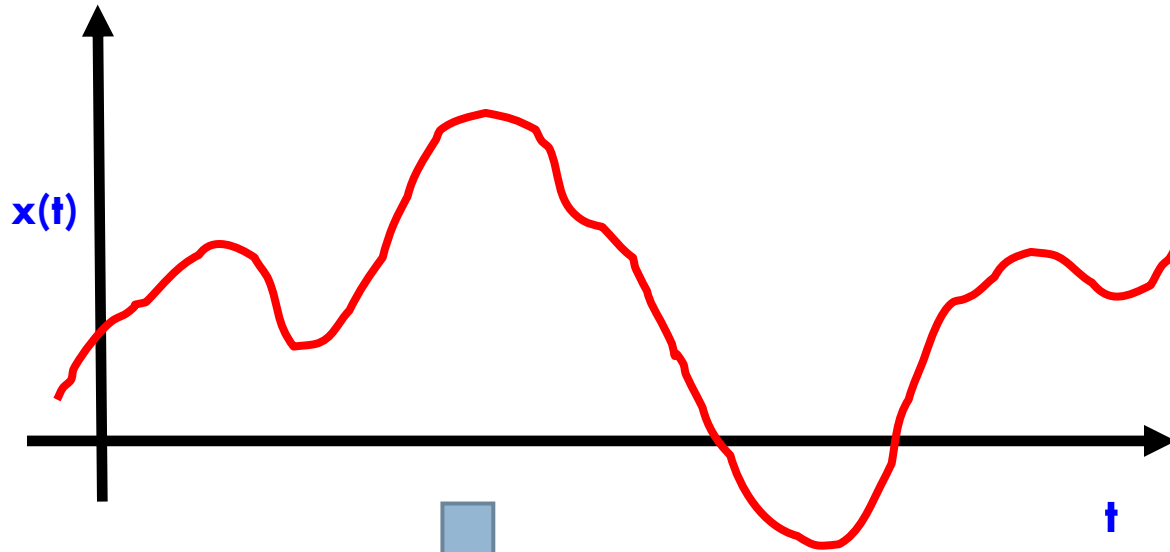
Sampling theorem

- A **continuous-time** signal $x(t)$ with a spectral band B strictly limited ($X(f) = 0$ for $|f| > \pm B$) can be uniquely reconstructed from its sampled version $x(n)$ ($n = 0, \pm 1, \pm 2, \pm 3, \dots$) if the sampling frequency $f_c = 1/T_c$ satisfies the following relation

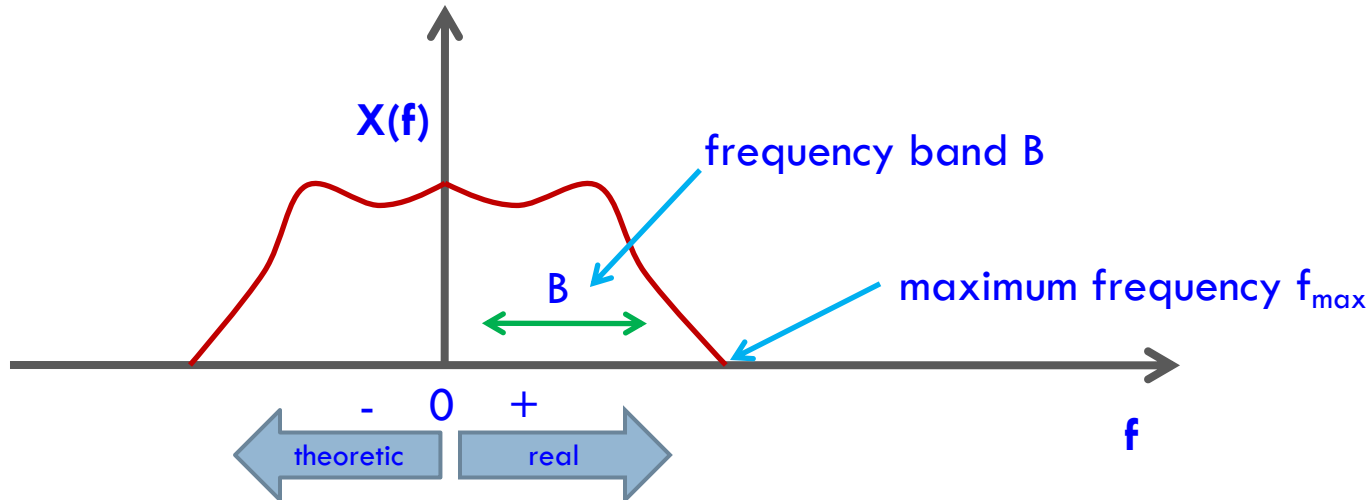
$$f_c = \frac{1}{T_c} \geq 2B$$



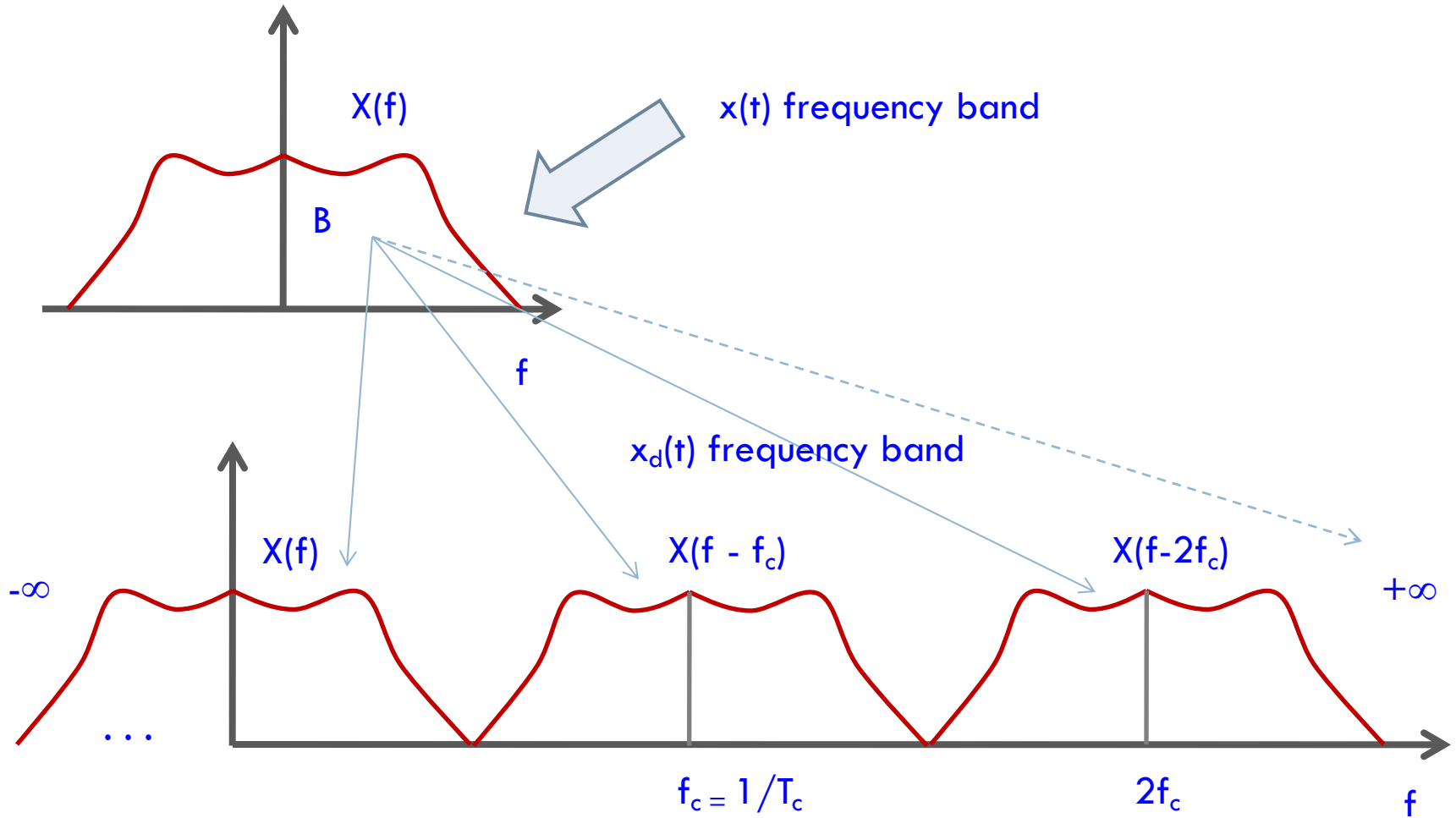
Sampling theorem



frequency domain transform

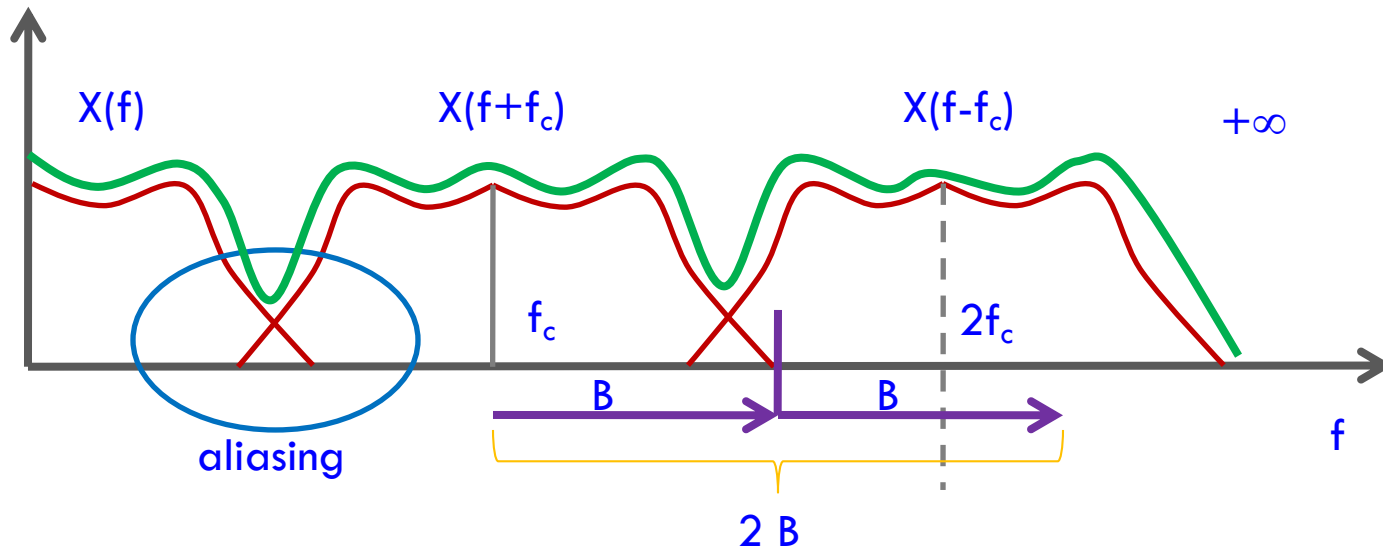


Sampling theorem



Sampling theorem

$$f_c < 2B$$

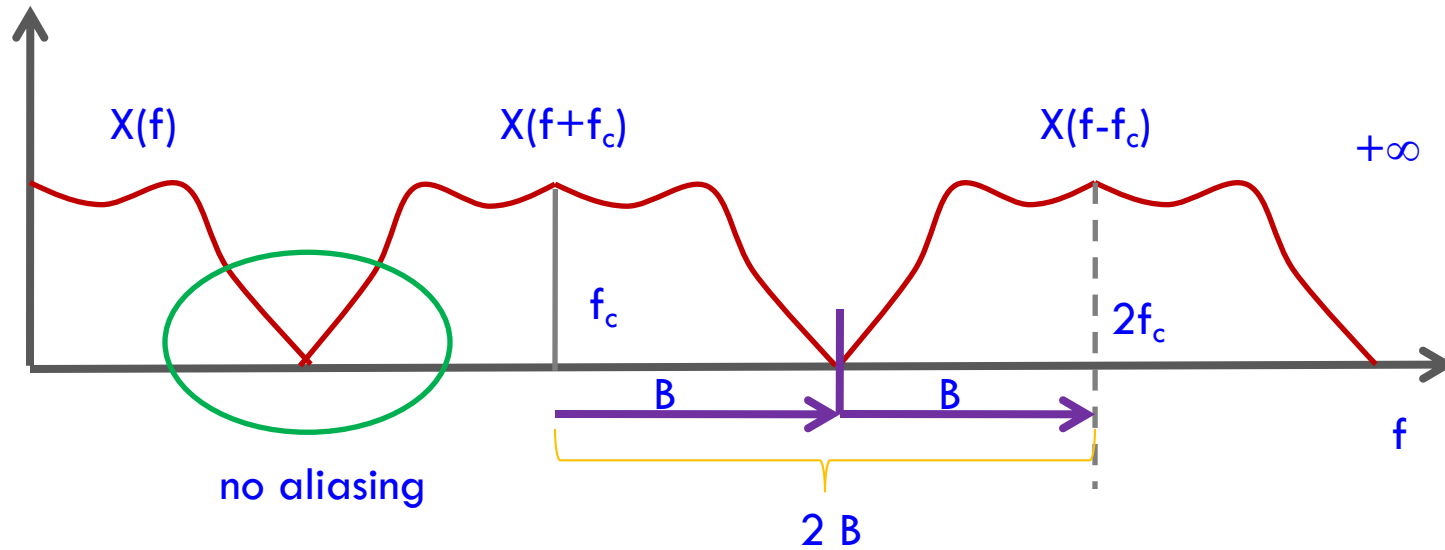


Subsampling of the signal



Sampling theorem

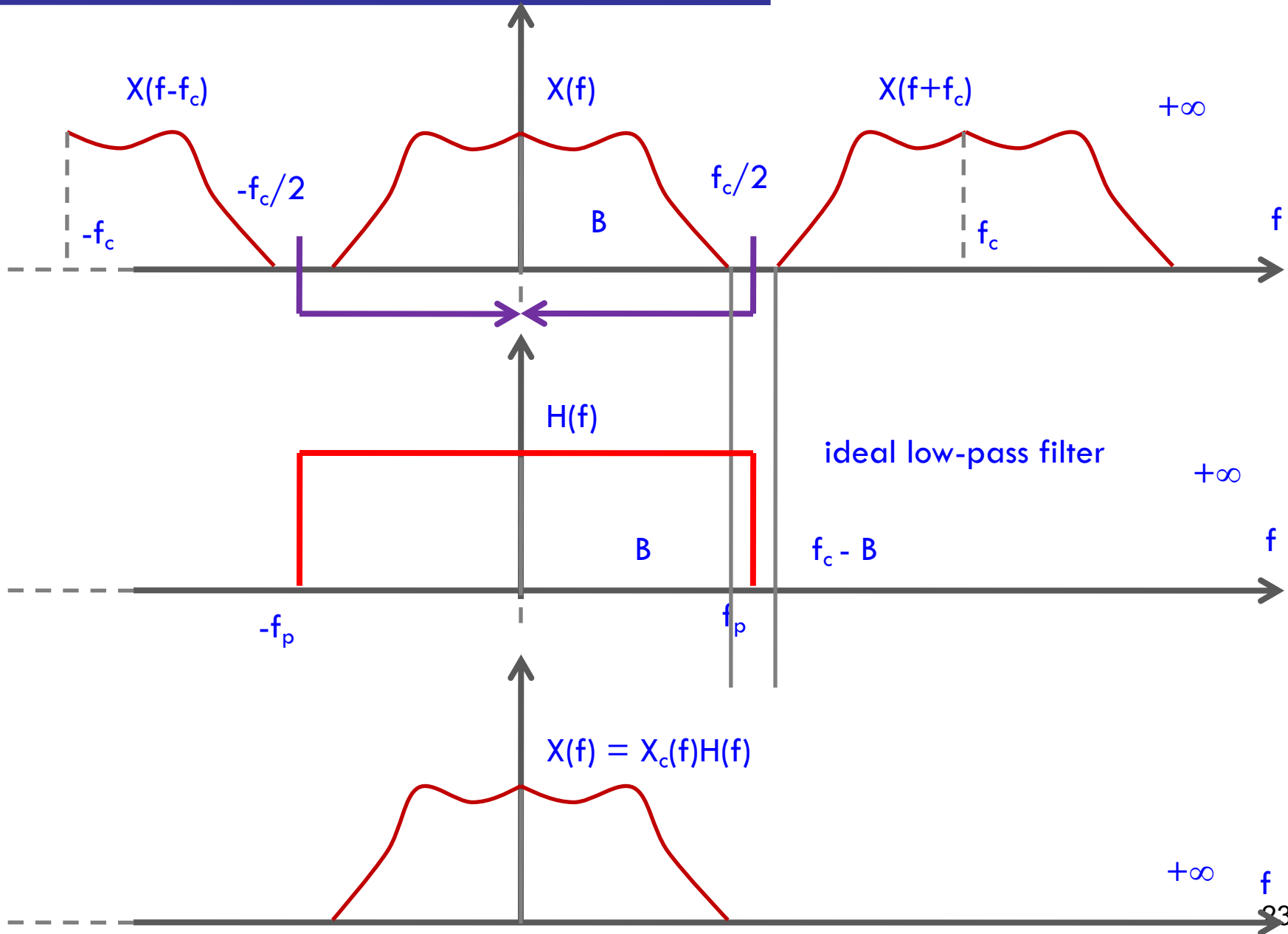
$$f_c \geq 2B$$



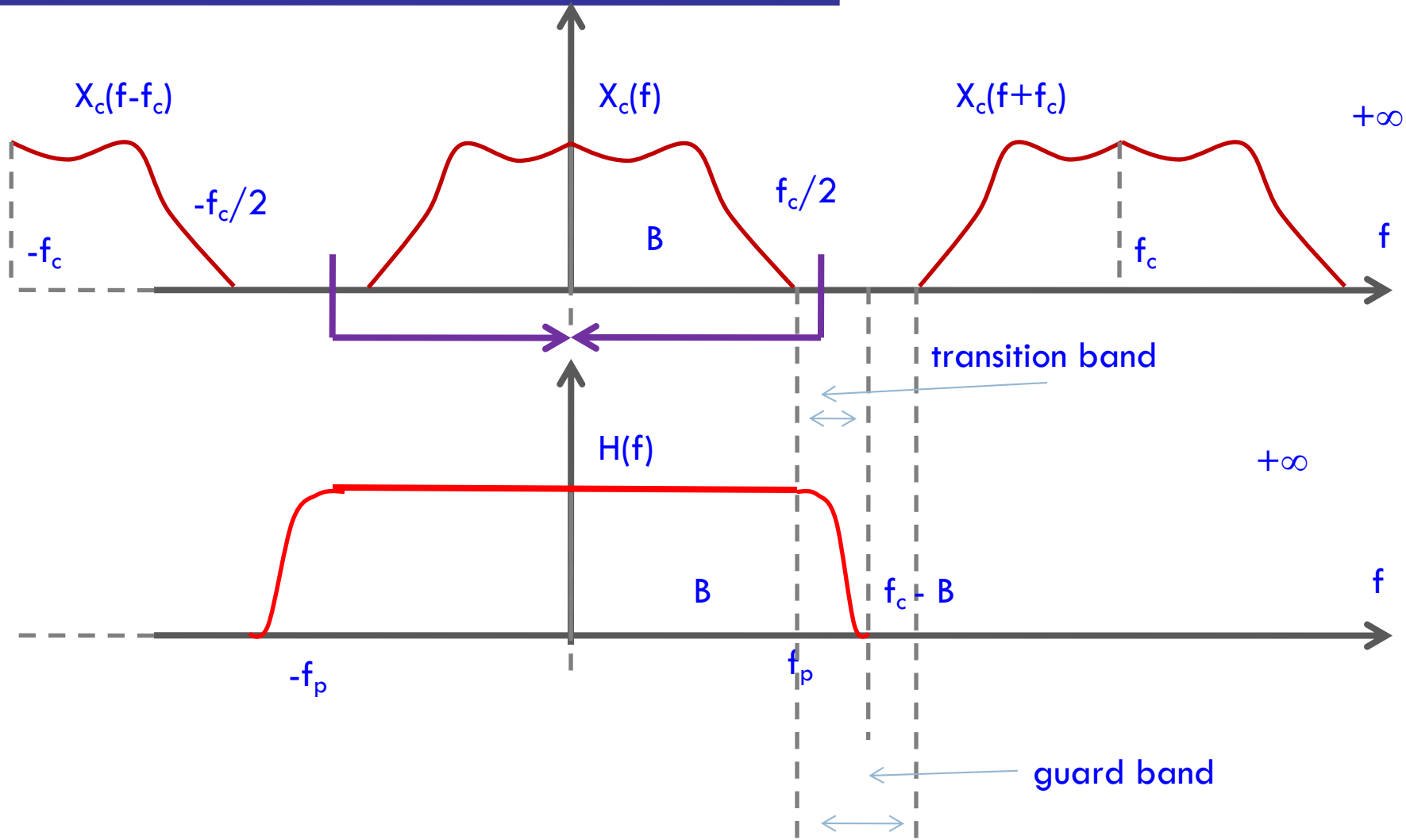
Oversampling of the signal



Signal reconstruction



Real filters

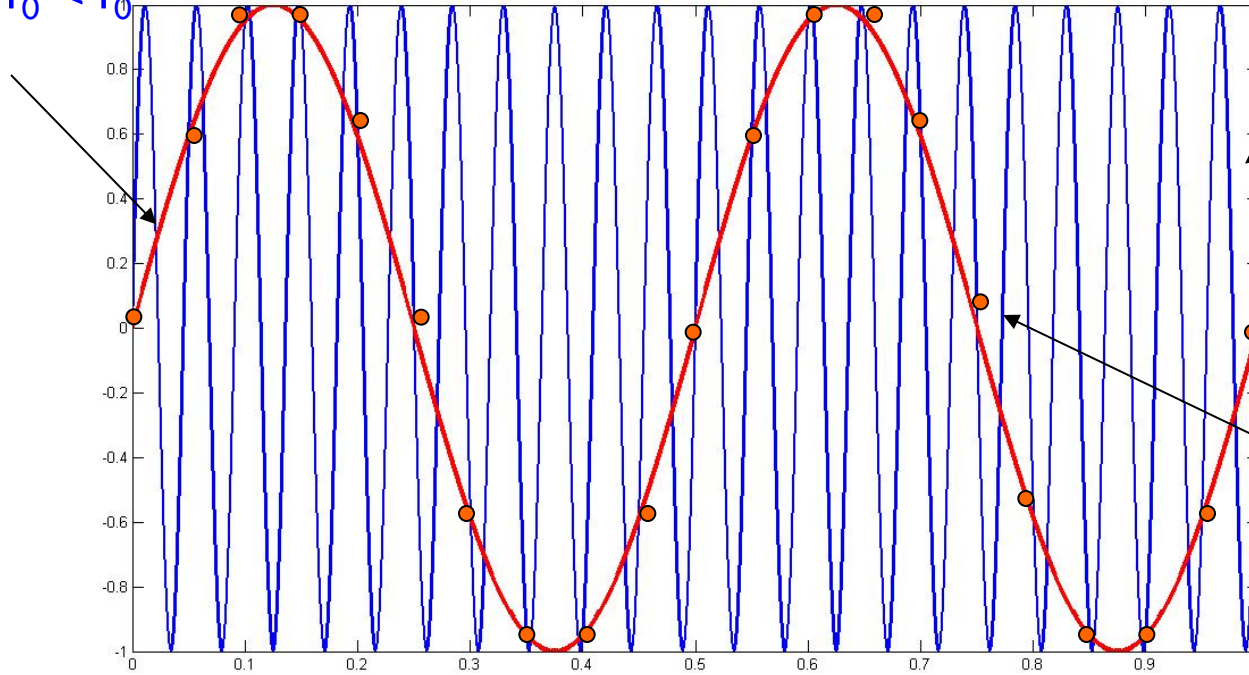


Example

Reconstruct signal

$$f_{\text{alias}} =$$

$$f_c - f_0 < f_0$$



Source signal with frequency f_0

Samples with $f_c < 2f_0$

$$f_{\text{alias}} = f_c - f_{\text{reale}} < f_{\text{reale}}$$

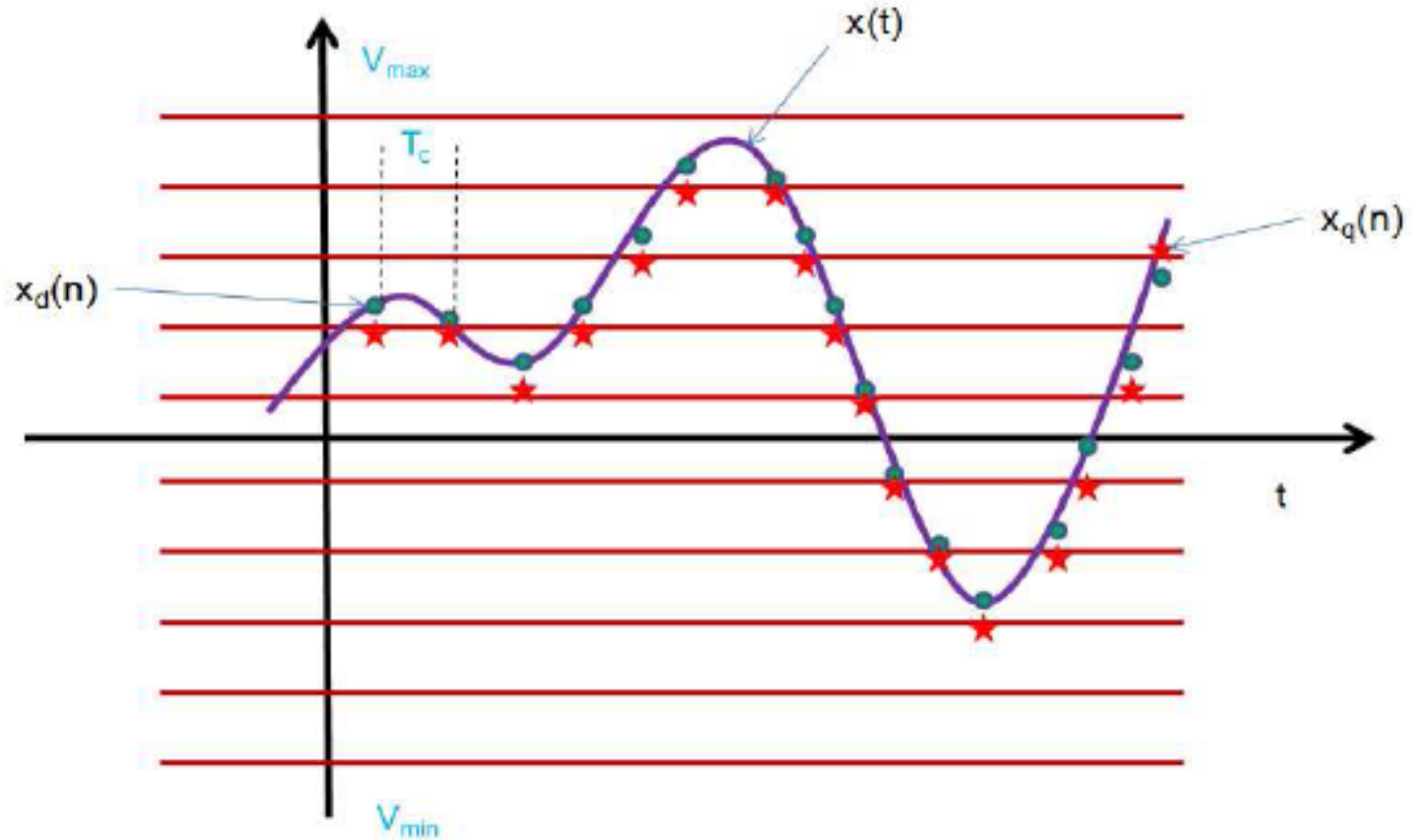


Quantization

- **Quantization** is the procedure of constraining something from a continuous set of values (such as the real numbers) to a relatively small **discrete set** (such as the integers)
- Quantization **replaces** each real number with an approximation from a finite set of discrete values (**levels**)
 - values are represented as fixed-point words or floating-point words
 - common **word-lengths** are 8-bit (256 levels), 16-bit (65,536 levels), 32-bit (4.3 billion levels)



Example of quantization

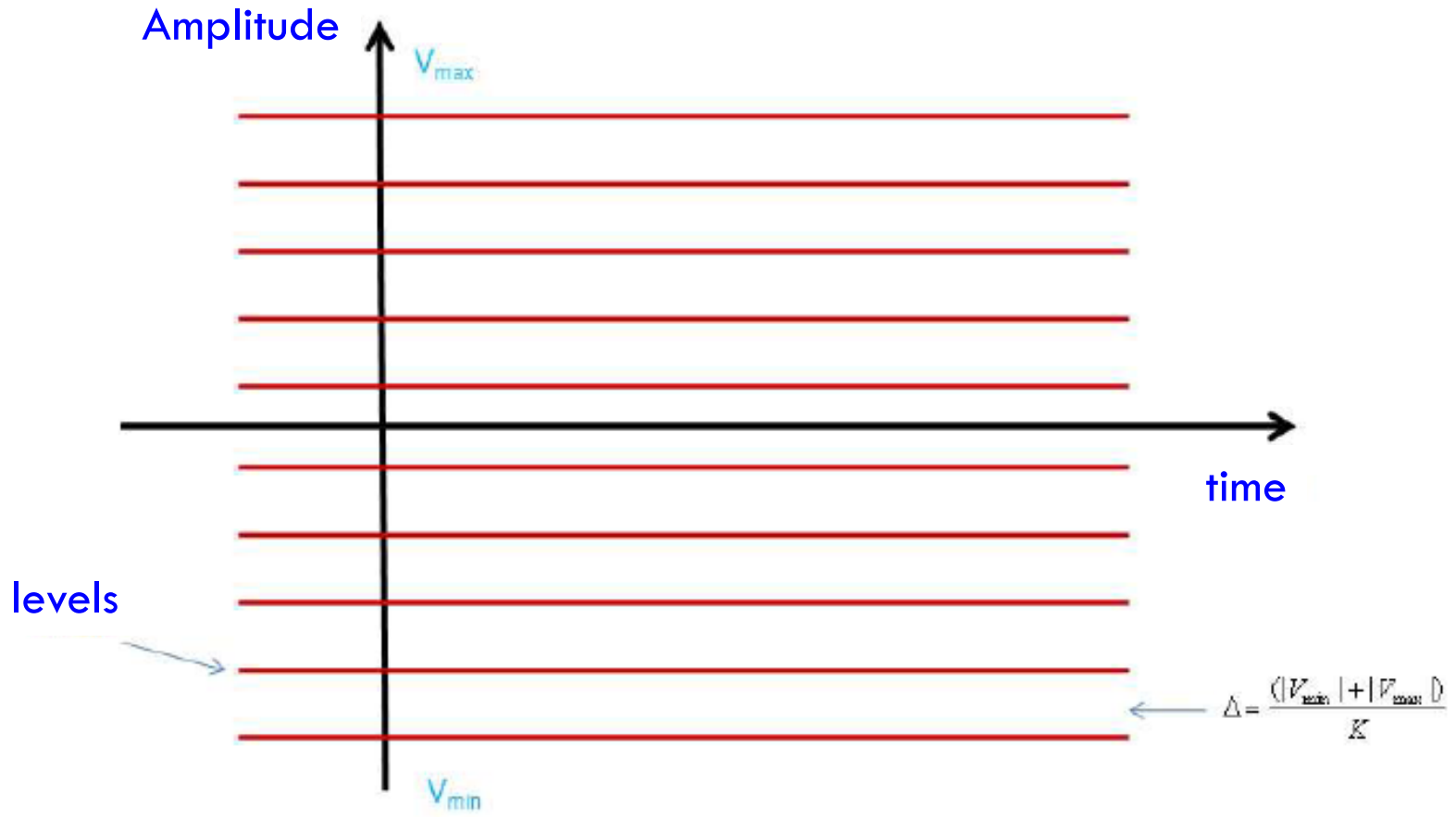


$$e(n) = x_q(n) - x_d(n)$$

quantization error



Coding

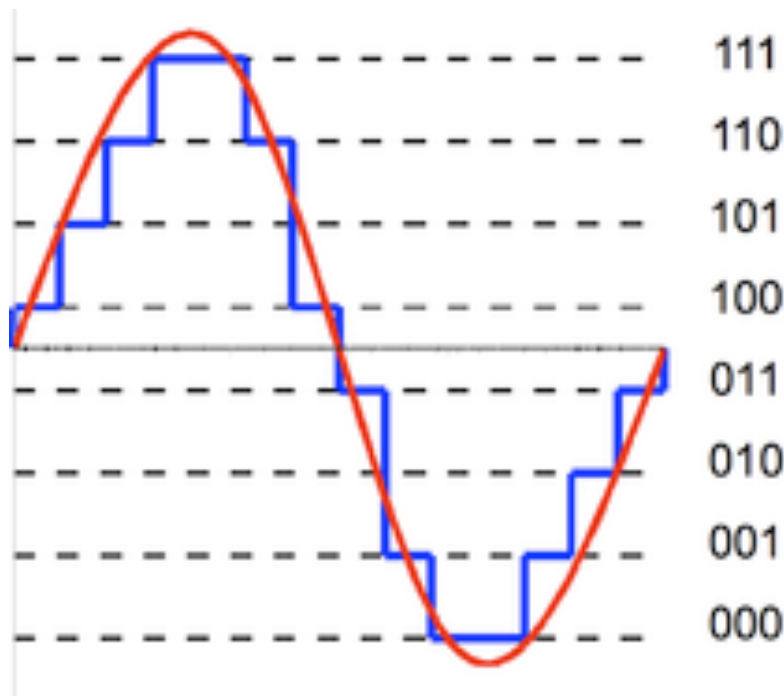


Uniform quantization



Coding

- With **N bits** $K = 2^N$ quantization levels are obtained
 - at each level a code of N bits can be associated



3-bit resolution with eight levels



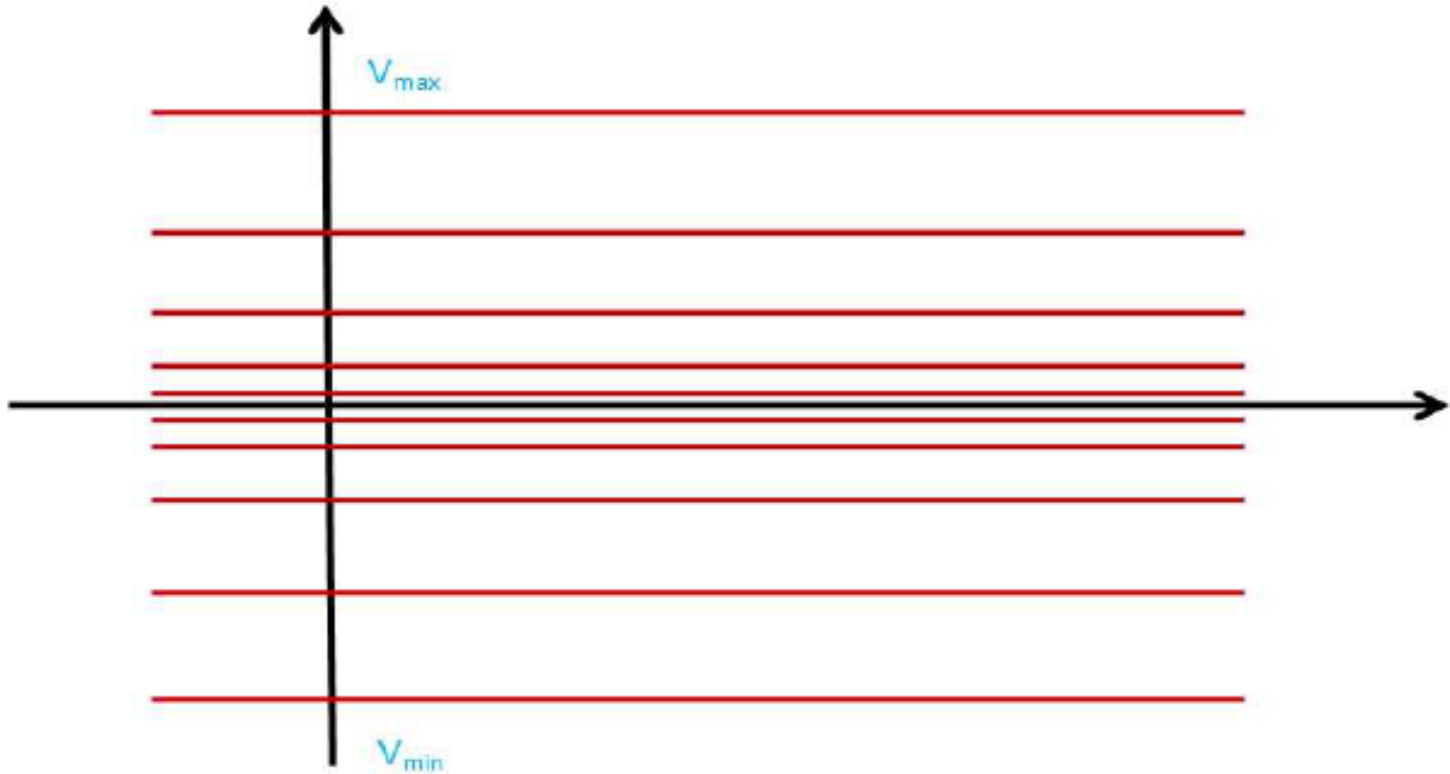
Bit rate

- Bit rate
 - number of bits per second
 - product between the **sampling frequency** and the **number of quantization bits**

$$\text{bit rate} = f_c \cdot N$$



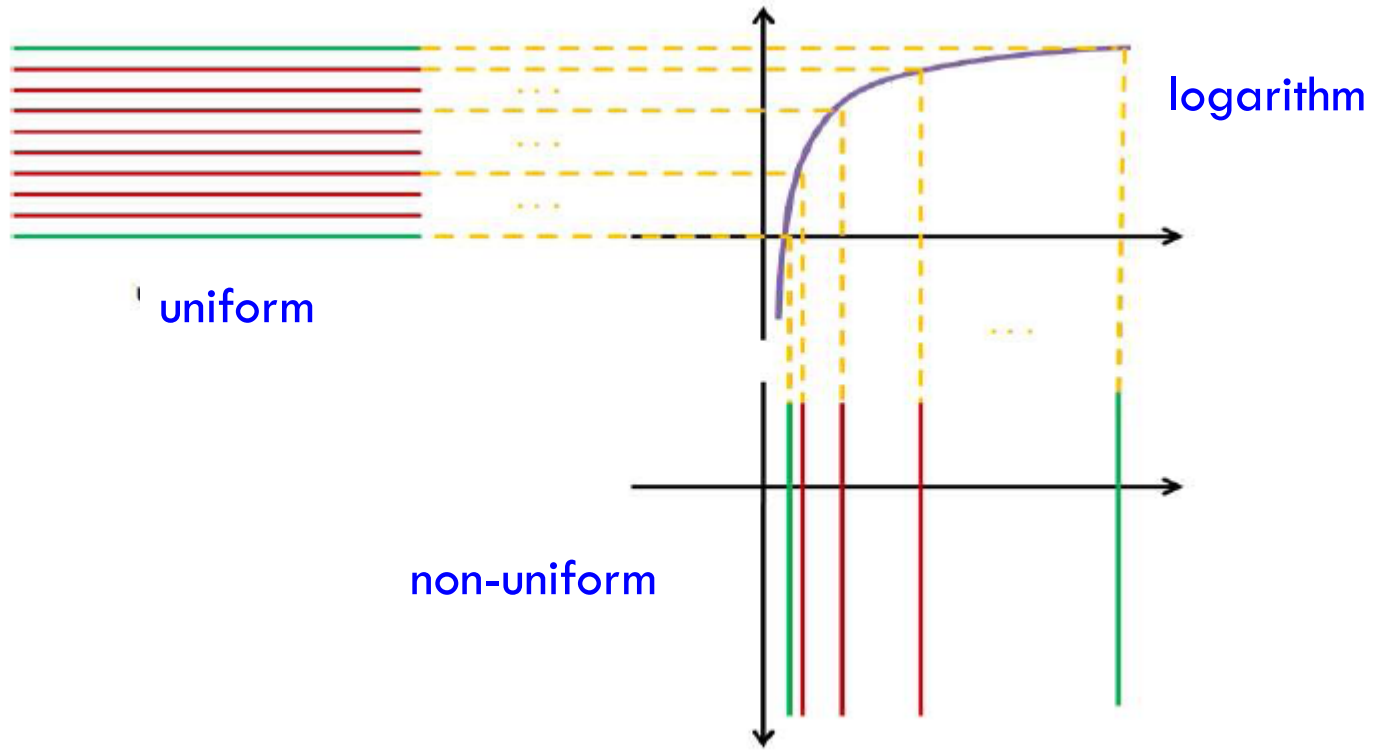
Non-linear quantization



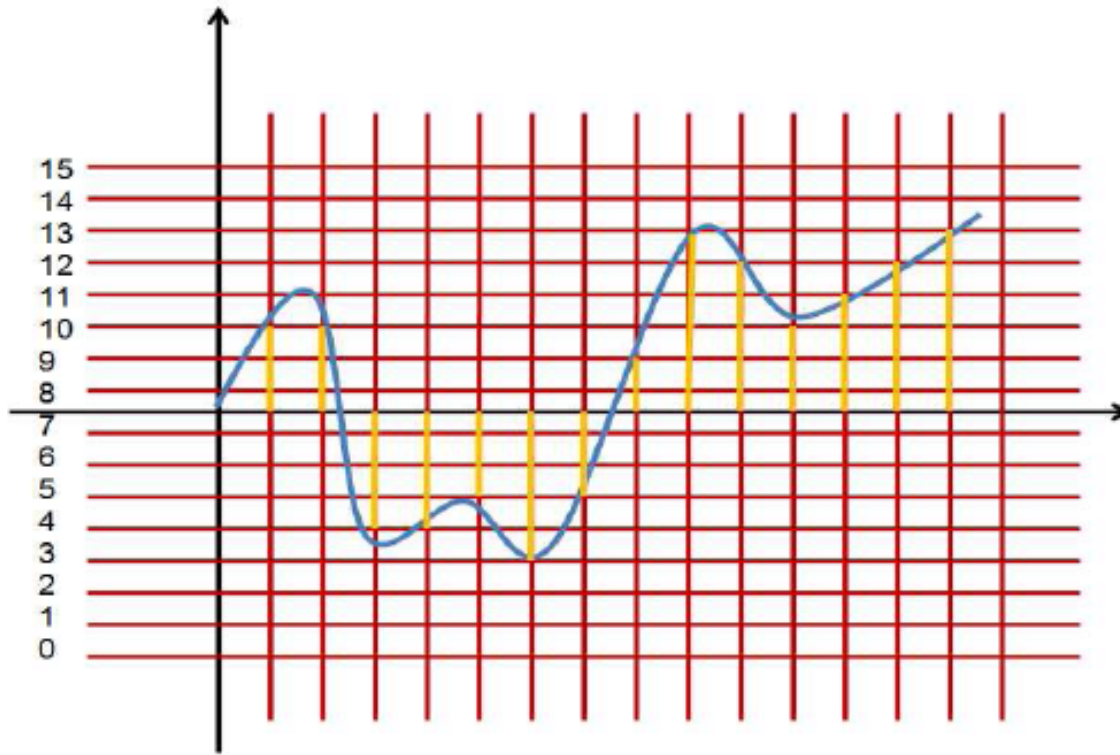
non-uniform quantization



Logarithmic quantization



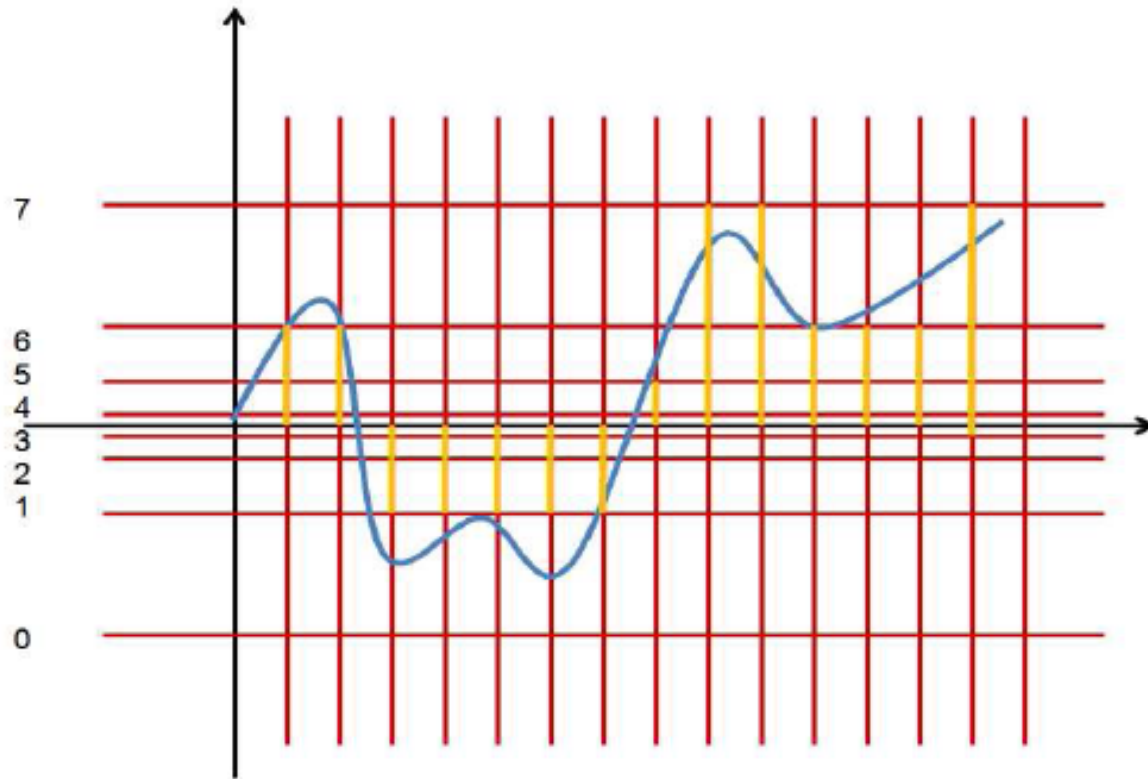
Example



Uniform quantization



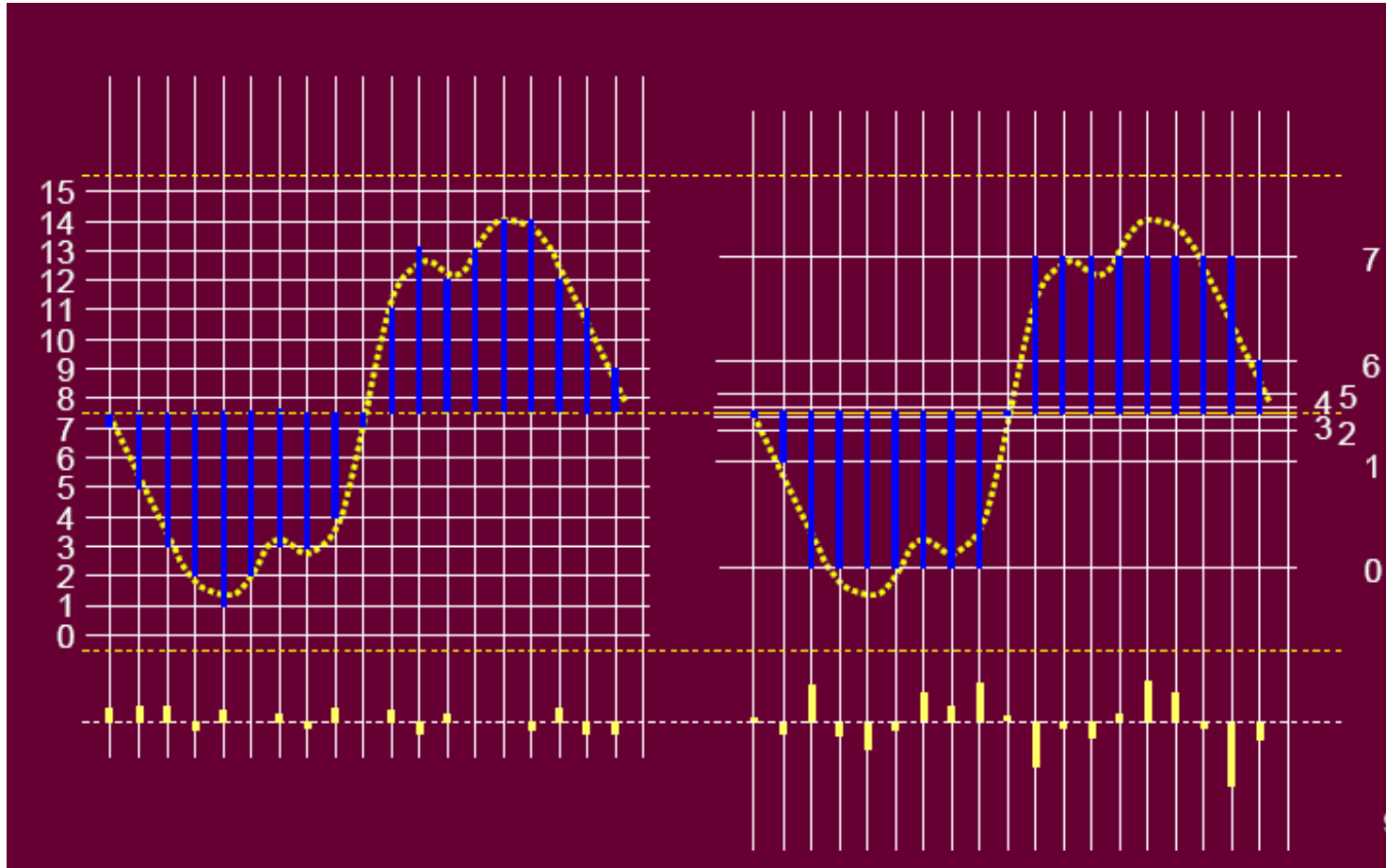
Example



non-uniform quantization



Comparison



4 bit linear

3 bit logarithmic



Comparison

■ Quality

- the dynamic range of the 8-bit logarithmic quantization corresponds to 13-14 bits linear quantizer

■ Signal to Noise Rate (SNR)

- a 8-bit logarithmic converter is better than a 8-bit linear at low amplitudes, but worse at high amplitudes



Python code (Tono_Puro.py)

```
import numpy as np

import sounddevice as sd

import matplotlib.pyplot as plt

# Signal frequency
frequency = 440

# Amplitude
amplitude = 100

# Sampling frequency
sampling_rate = 44100

# Seconds of the sequence
sec = 1

# Number of samples
num_samples = sec * sampling_rate

# time
t = np.arange(0, num_samples-1)

# sinusoidal wave
sine_wave = amplitude*np.sin(2 * np.pi * frequency * t / sampling_rate)
```



Python code

```
# Function plot
plt.plot(t/sampling_rate, sine_wave, color='blue', label="tono puro")
plt.title('Tono Puro')
plt.xlabel('t')
plt.ylabel('y(t)')
plt.legend()
plt.show()

# Sound of the signal
sd.play(sine_wave, sampling_rate)
sd.stop()
```



Python IDE: IDLE

```
Python 3.7.2 Shell
Python 3.7.2 (v3.7.2:9a3ffc0492, Dec 24 2018, 02:44:43)
[Clang 6.0 (clang-600.0.57)] on darwin
Type "help", "copyright", "credits" or "license()" for more information.
>>>

Tono_Puro.py - /Users/angelociaramella/Documents/Documents/Lectures/A A 201...
#!/usr/bin/env python3
# -*- coding: utf-8 -*-
"""
Created on Sat Mar  9 09:16:02 2019

@author: angelociaramella
"""
from typing import List
import numpy as np
import matplotlib.pyplot as plt

# frequency is the number of times a wave repeats a second
frequency = 440 # 1000
sampling_rate = 44100 # 48000.0
sec = 1 # seconds
num_samples = sec * sampling_rate # 48000

# The sampling rate of the analog to digital convert
amplitude = 100 # 16000

file = "test.wav"

sine_wave = [amplitude*np.sin(2 * np.pi * frequency * x / sampling_rate) for x in range(num_samples)]

nframes = num_samples
comptype = "NONE"
comprname = "not compressed"

Ln: 4 Col: 4
Ln: 1 Col: 0
```



Python IDE: PyCharm

The screenshot displays the PyCharm IDE interface. The main editor window shows a Python script named `Tono_Puro.py` with the following code:

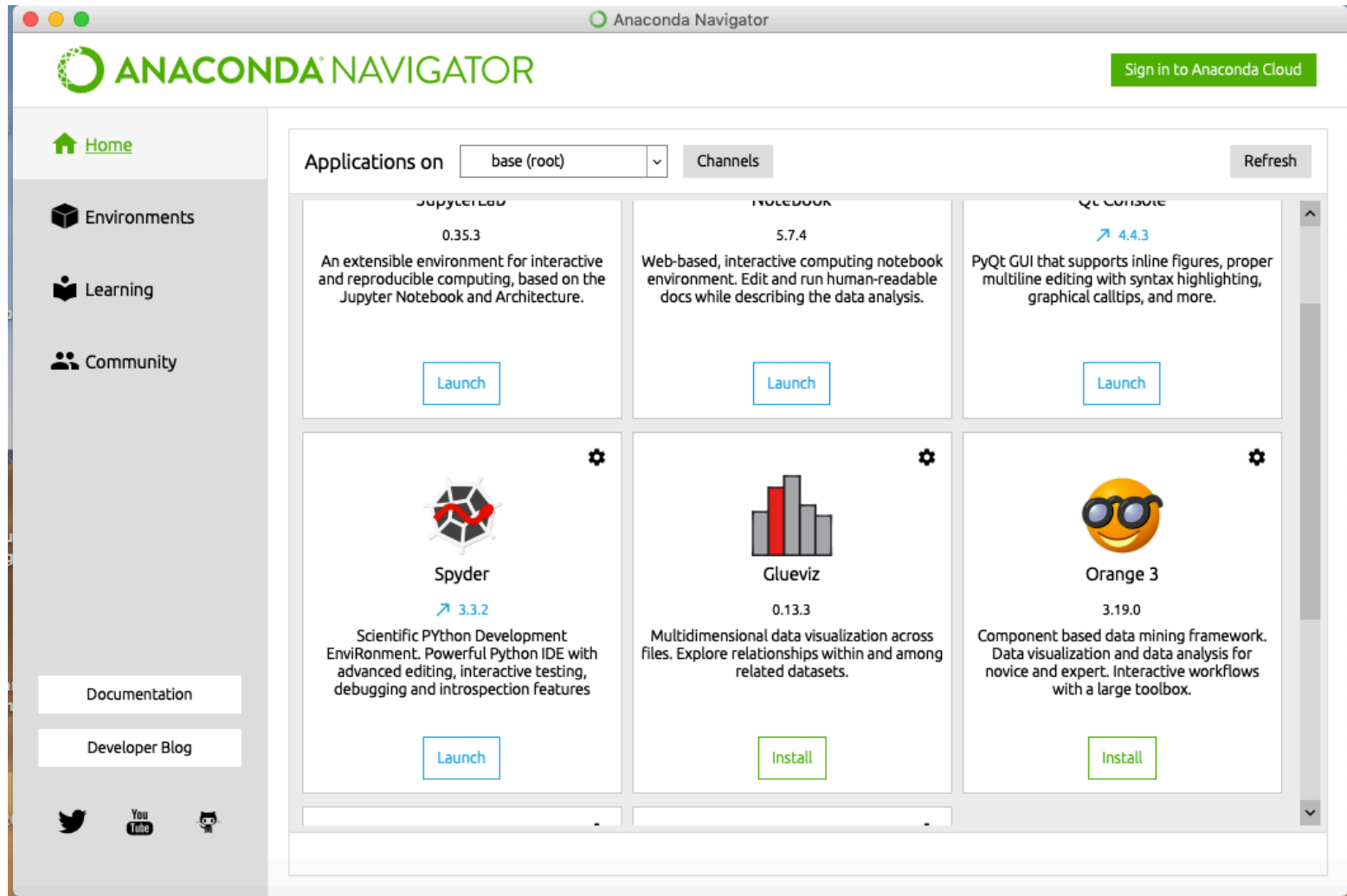
```
24 sampling_rate = 44100
25 # Seconds of the sequence
26 sec = 1
27
28 # Number of samples
29 num_samples = sec * sampling_rate
30
31 # time
32 t = np.arange(0, num_samples-1)
33
34 # sinusoidal wave
35 sine_wave = amplitude * np.sin(2 * np.pi * frequency * t / sampling_rate)
36
37 # Function plot
38 plt.plot(t, sine_wave, color='blue', label="tono puro")
39 plt.title('Tono Puro')
40 plt.xlabel('t')
41 plt.ylabel('y(t)')
42 plt.legend()
43 plt.show()
44
45 # Sound of the signal
46 sd.play(sine_wave, sampling_rate)
47 sd.stop()
48
```

The Run console at the bottom shows the command used to execute the script:

```
/Users/angelociaramella/Tono_Puro_1/bin/python /Users/angelociaramella/PycharmPro...
```

On the right side, a window titled "Figure 1" displays a plot of the sine wave. The plot is titled "Tono Puro" and shows a blue line representing the signal. The x-axis is labeled "t" and ranges from 0 to 40,000. The y-axis is labeled "y(t)" and ranges from -100 to 100. A legend in the bottom right corner identifies the blue line as "tono puro".

Python IDE: Anaconda



Python IDE: Spider

The screenshot displays the Spyder Python IDE interface. The left pane shows the editor with the following Python code:

```
1#!/usr/bin/env python3
2# -*- coding: utf-8 -*-
3"""
4Created on Sat Mar 9 09:16:02 2019
5
6@author: angelociaramella
7"""
8from typing import List
9
10import numpy as np
11
12#import wave
13
14#import struct
15
16import matplotlib.pyplot as plt
17
18
19# frequency is the number of times a wave repeats a second
20
21frequency = 440 # 1000
22
23sampling_rate = 44100 # 48000.0
24
25sec = 1 # seconds
26
27num_samples = sec * sampling_rate # 48000
28
29# The sampling rate of the analog to digital convert
30
31amplitude = 100 # 16000
32
33file = "test.wav"
34
35
36sine_wave = [amplitude*np.sin(2 * np.pi * frequency * x / sampling_rat
37
38
39
40nframes = num_samples
41
42comptype = "NONE"
```

The right pane shows the IPython console with the following output:

```
In [2]:
In [2]: runfile('/Users/angelociaramella/Documents/Documents/Lectures/A A 2018-2019/II semester/Intelligent Signal Processing/python-machine-learning-book-master/ML/Tono_Puro.py', wdir='/Users/angelociaramella/Documents/Documents/Lectures/A A 2018-2019/II semester/Intelligent Signal Processing/python-machine-learning-book-master/ML')
```

Below the console, a plot titled "Tono Puro" is displayed. The plot shows a sine wave with the following characteristics:

- Y-axis: $y(t)$, ranging from -100 to 100.
- X-axis: t , ranging from 0 to 40000.
- Legend: "tono puro".

The status bar at the bottom indicates: Permissions: RW, End-of-lines: LF, Encoding: UTF-8, Line: 71, Column: 2, Memory: 62 %.

Homework

- Python code
 - Implementing a class by code on the Tono_Puro.py

