

# Audio Digitale

---

Elaborazione dell'audio digitale  
*Ingegneria del Cinema, Informatica e Telecomunicazioni*

**Antonio Servetti**

Internet Media Group

Dip. di Automatica ed Informatica

Politecnico di Torino

[servetti@polito.it](mailto:servetti@polito.it)

<http://media.polito.it>

# Sommario

---

- Introduzione all'audio digitale
  - Mixing e routing (base)
  - Quantizzazione (e cenni di campionamento)
    - ✓ Numero bit ( $N$ ), intervallo di quantizzazione  $[-X_m, +X_m]$ , passo di quantizzazione  $Q$ , valori di quantizzazione  $q_k$
  - Audio digitale con Python
  - Formati audio
  - Analisi dell'errore di quantizzazione
    - ✓ Signal to (Quantization) Noise Ratio (SNR, SQNR)
    - ✓ Distorsione armonica: clipping, low-level distortion
- } PARENTESI

# Riferimenti

---

- [VMM] "Voicemeeter – User Manual"  
[https://vb-audio.com/Voicemeeter/VoicemeeterBanana\\_UserManual.pdf](https://vb-audio.com/Voicemeeter/VoicemeeterBanana_UserManual.pdf)
- [VALLE] Lombardo, Valle, "Audio e Multimedia – 5e", 2024
- [POHL] Pohlmann, "Principles of digital audio – 6e", 2010
- [FMP] Python Notebooks for Fundamentals of Music Processing  
<https://www.audiolabs-erlangen.de/resources/MIR/FMP/B/B.html>

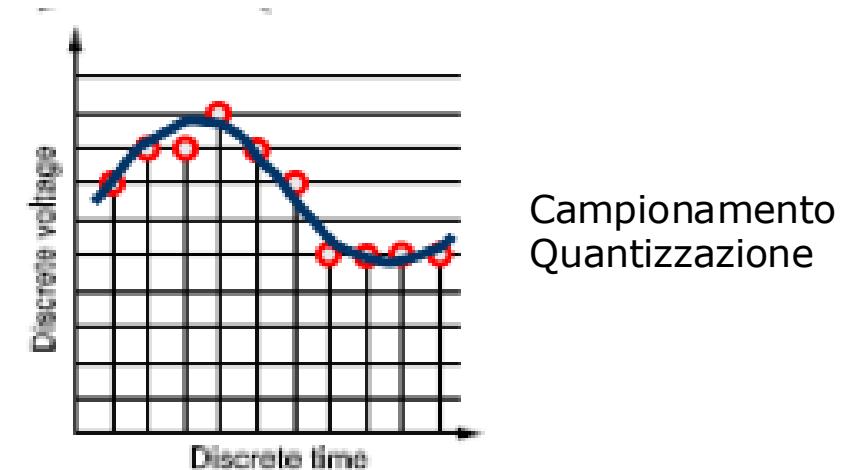
# Riferimenti

---

- Introduzione all'audio digitale – [VALLE 3.1]
- Mixing e routing (base) [VMM]
- Quantizzazione (e cenni di campionamento) [VALLE 3.2, 3.3]
- Audio digitale con Python [FMP - Basics]
- Formati audio [VALLE 8.1]
- Analisi dell'errore di quantizzazione [POHL 2.]
  - ✓ Signal to (Quantization) Noise Ratio (SNR, SQNR)
  - ✓ Distorsione armonica: clipping, low-level distortion

# Da analogico a digitale

- Premessa: già avete sentito di rappresentazione digitale dei segnali
  - ✓ Analogica: rappresentazione continua (nel tempo e nel valore)
  - ✓ Digitale: rappresentazione discreta (nel tempo e nel valore)



- Quali sono i vantaggi e quali gli svantaggi?

# Vantaggi (alcuni)

- Non vi è più degradazione nella copia o nella trasmissione: *la qualità è funzione soltanto del processo di conversione*
- Non sono più necessari equipaggiamenti dedicati (e costosi) per l'elaborazione audio: basta un calcolatore
- Archiviazione indipendente dal supporto: non più nastro, vinile, ...
- Elaborazione numerica molto più semplice: compressione, effetti, sintesi



Prima e dopo



Reference: Watkinson, "The art of digital audio", Ch1 Introducing digital audio

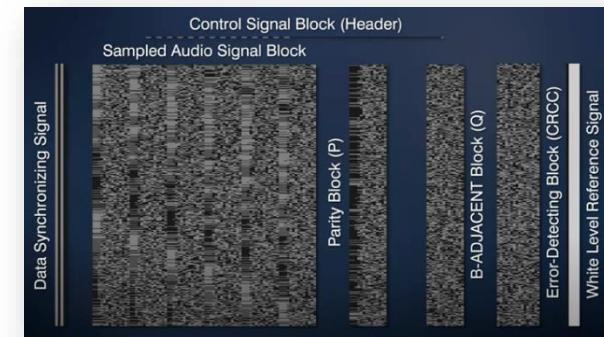
# Svantaggi

---

- Cura nella conversione da analogico a digitale
  - ✓ Attenzione ad aliasing, distorsioni e volume (vedremo in seguito)
- Ridotte capacità di calcolo
  - ✓ PCM brevettato dall'inglese Alex Reeves nel 1938, si è dovuto aspettare il transistor negli anni '50 (e più)
- Elevati requisiti di memoria per l'archiviazione e la trasmissione
  - ✓ PCM 16 bit richiede una banda analogica di 1-1.5 MHz (S/PDIF 3-g MHz), enorme cfr 15-20 kHz richiesti per la registrazione analogica
  - ✓ 1957 Max Mathews e Guttman (Bell Labs) creano il primo brano musicale generato dal calcolatore (17 secondi)
  - ✓ Necessità di compressione (telefonia, mp3, streaming)

# Perché 44.1 kHz

- La frequenza di campionamento 44.1 kHz (CD-ROM) nasce da una necessità che risale agli anni '70
  - ✓ I registratori video erano gli unici con banda sufficiente (MHz) per registrare audio digitale
  - ✓ Venivano utilizzati dei dispositivi "PCM Adaptor" per convertire i bit in blocchi bianchi (0) o neri (1)
- Registrazioni su sistemi PAL ed NTSC
  - ✓ PAL 294 linee interlacciate a 25 fps
  - ✓ NTSC 245 linee interlacciate a 30 fps
  - ✓ A circa 13 bit per campione, 3 campioni per linea
  - ✓  $3 \text{ campioni} * 294 \text{ (o 245) linee} * 2 \text{ interlacciate} * 25 \text{ (o 30) frame al secondo} = 44.1 \text{ kHz}$



Youtube: Mw3IZ\_y1DIw

Reference: [https://en.wikipedia.org/wiki/44,100\\_Hz](https://en.wikipedia.org/wiki/44,100_Hz)

# Perché 48 kHz

---

- In ambito A/V è invece prevalsa la scelta per 48 kHz prevalentemente a motivo del fatto che
  - ✓ I film (movies) venivano registrati a 24 fps (rapporto 2:1)
  - ✓ L'European Broadcasting Union aveva standardizzato i 32 kHz per la radio, già usati dalla BBC (rapporto intero 3:2)
  - ✓ Un multiplo intero dei 30 fps di NTSC, e.g. 60 kHz, era eccessivo

**"The speed of the film bearing the sound track had been standardized at 90 feet a minute, 24 frames a second.** Although tradition has had it that 90 feet a minute was the optimum speed for the quality of sound reproduction, the fact of the matter was that, originally, Earl Sponable and Theodore Case had been experimenting with a speed of 85 feet a minute, which appears to have worked satisfactorily. As Sponable said, "After our affiliation with the [Fox company] this was changed to ninety feet a minute in order to use the controlled motors already worked out and used in the Vitaphone system." The standardization, then, was not made for purposes of sound quality, but for maximum profit for Western Electric."

*"The Speed of Sound, Hollywood and the Talkie Revolution", by Scott Eyman*

N.B. I film muti venivano registrati (per ridurre i costi) a 16-18 fps. Vitaphone è stato il primo sistema per associare una traccia audio sincronizzata. "The Jazz Singer" (1927) è il film che segna la nascita del cinema sonoro.

---

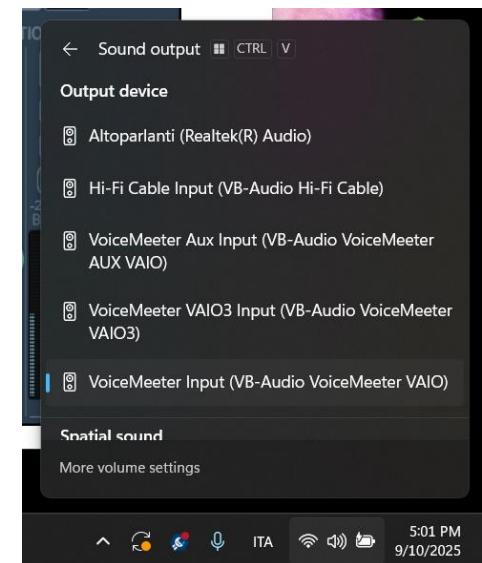
Reference: [https://en.wikipedia.org/wiki/48,000\\_Hz](https://en.wikipedia.org/wiki/48,000_Hz)

---

## MIXING e ROUTING - base

# Ingressi e uscite

- Ingresso: da dove proviene (viene acquisito) il suono
- Uscita: dove viene inviato (riprodotto) il suono
  
- Possono essere reali (fisici) o virtuali (software)
  - ✓ Reali: jack analogico (i/o), usb (i/o) o hdmi (o) digitale
  - ✓ Virtuali: voicemeeter, vb-audio, zoom, blackhole, ecc.
  
- Esempio: cattura suoni di sistema
  - ✓ Impostare l'uscita audio di default su Voicemeeter Input (VAIO)



# Dispositivi e driver

- L'elenco degli ingressi e delle uscite è costituito dai vari dispositivi fisici (Realtek) e virtuali (VB-Audio) collegati al sistema
- L'accesso ai dispositivi è reso possibile dai driver (driver diversi hanno prestazioni diverse)
- Driver Windows
  - ✓ MME 1991 (legacy): ampia compatibilità, elevato ritardo ( $> 100$  ms)
  - ✓ KS: kernel streaming, limitata compatibilità
  - ✓ WDM (WASAPI) 2007: alte prestazioni  
ritardo  $>30$ ms
  - ✓ ASIO 1997: Audio Streaming I/O (Steinberg)  
professionale (scheda audio dedicata o  
emulata da ASIO4ALL)



# Mixing e Routing

---

- Cosa si intende per mixing e routing
- Definizione: combinare (cioè sommare) più segnali audio tra loro (ingressi) in uno o più segnali nuovi (uscite)
  - ✓ Eventualmente con una elaborazione intermedia, ad esempio, controllo del volume (livello) – cioè del fattore di moltiplicazione con cui vengono sommati

Definiamo N ingressi x, ed M uscite y

$$y_j[n] = \sum_{i=1}^N r_{i,j} \cdot g_i \cdot x_i[n]$$

- $x_i[n]$  ingresso i-esimo (1..N)
- $g_i$  guadagno del canale i-esimo ( $0 \leq g \leq 4$ )
- $r_{i,m}$  coefficiente di routing (0 o 1)
- $y_j[n]$  uscita j-esima (1..M)

# Mixing e Routing

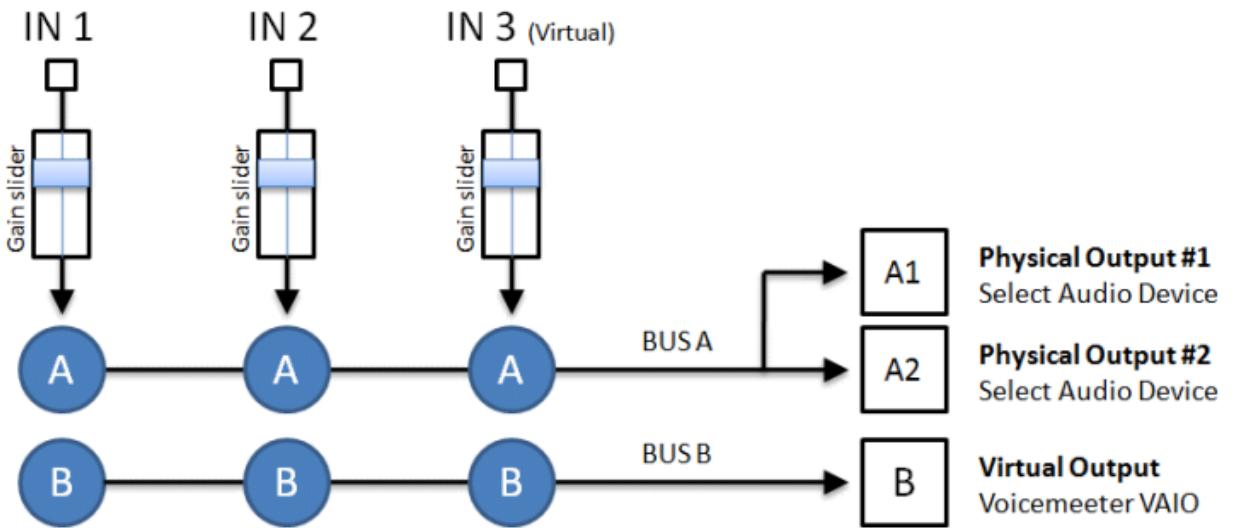


In VoiceMeeter il routing è implementato tramite BUS (sommatori).

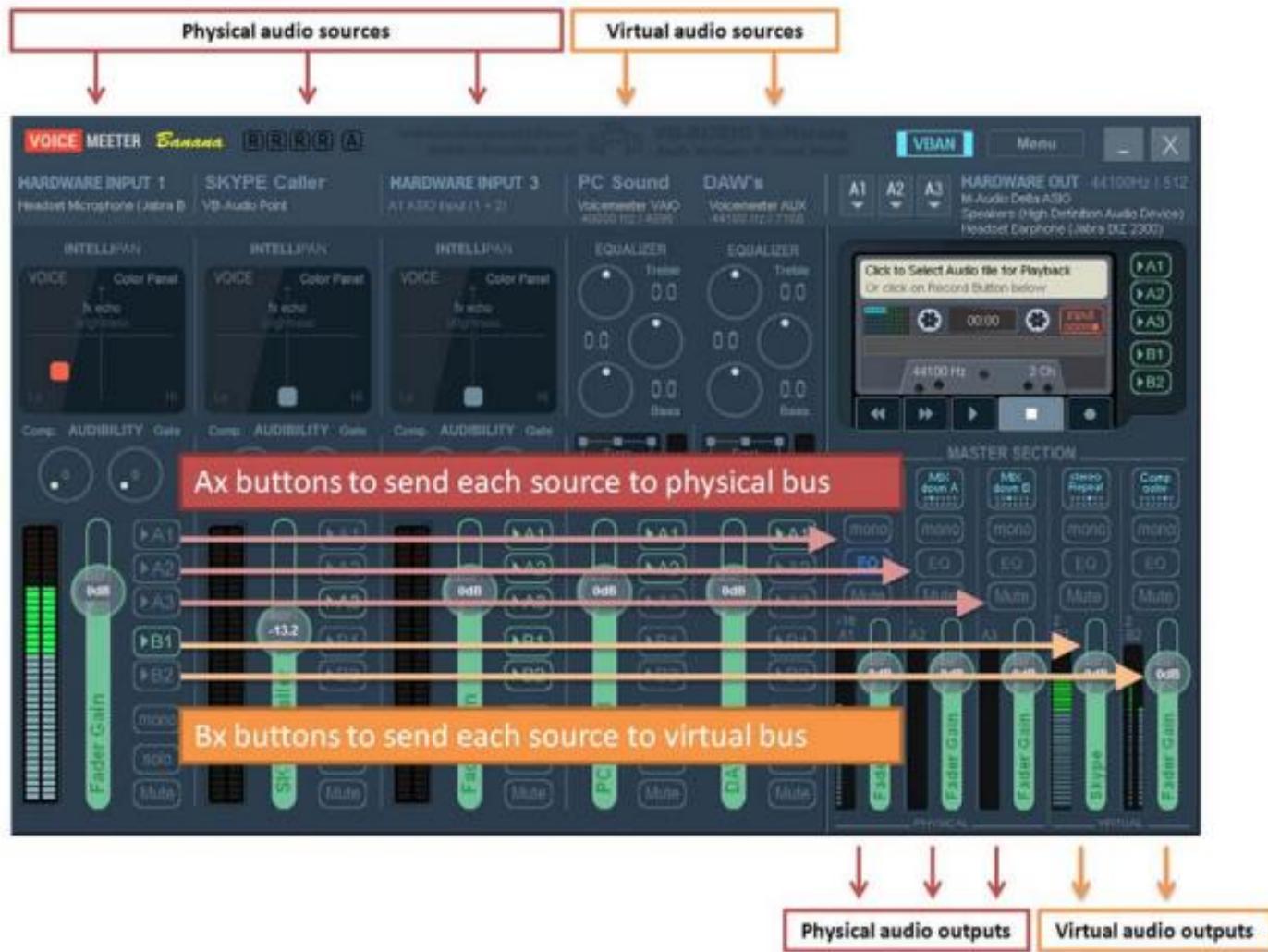
Ogni input ha dei bottoni mappati sulle uscite fisiche ( $A_j$ ) e su quelle virtuali ( $B_j$ ).

Il coefficiente di routing  $r_{ij}$  è 1/0 a seconda che il bottone sia selezionato o no.

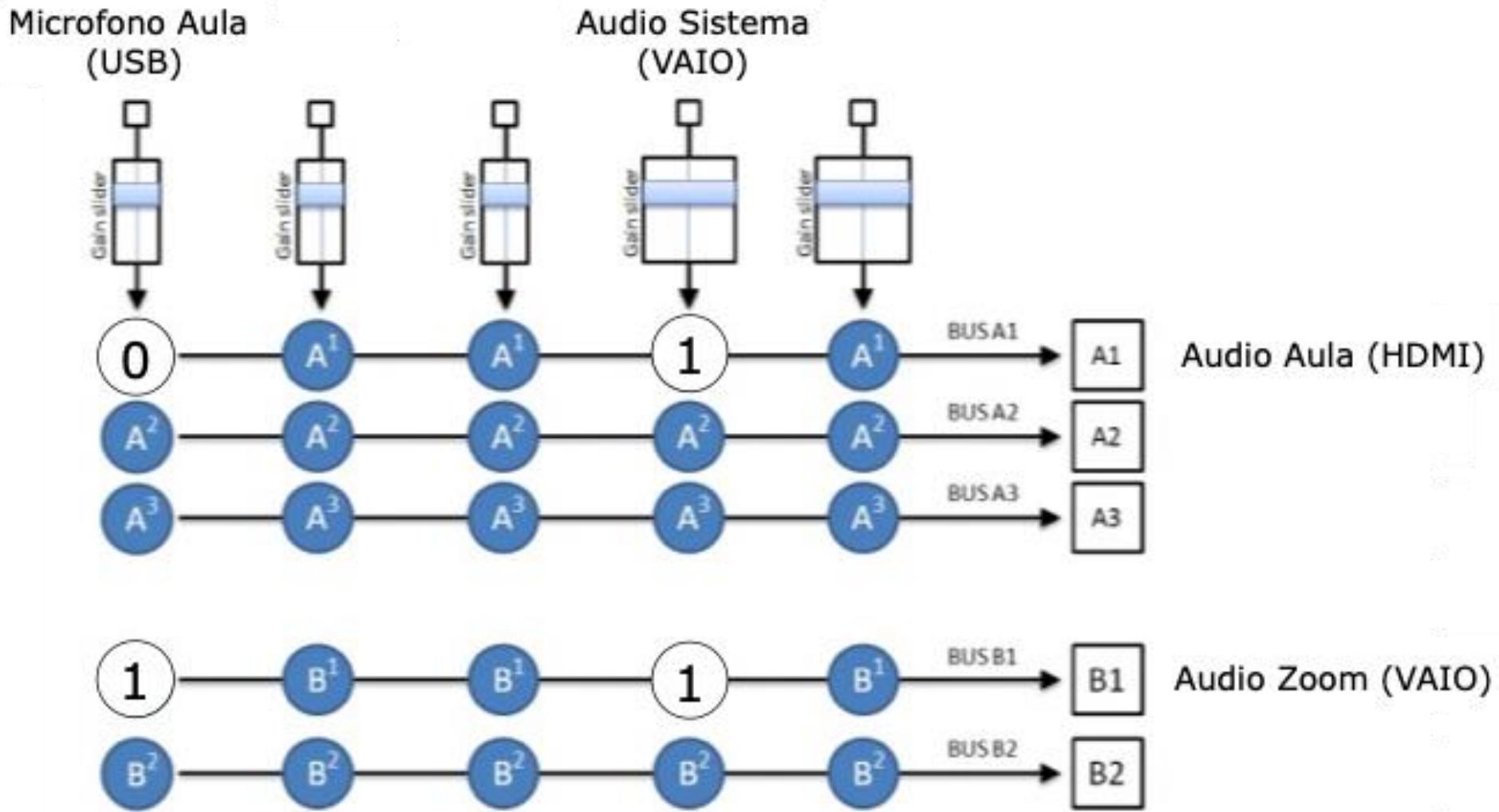
$$y_j[n] = \sum_{i=1}^N r_{i,j} \cdot g_i \cdot x_i[n]$$



# Mixing e Routing



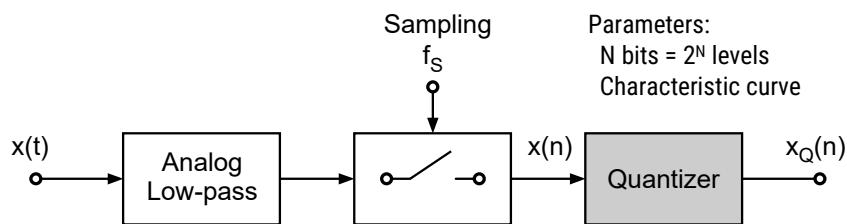
# Setup aula



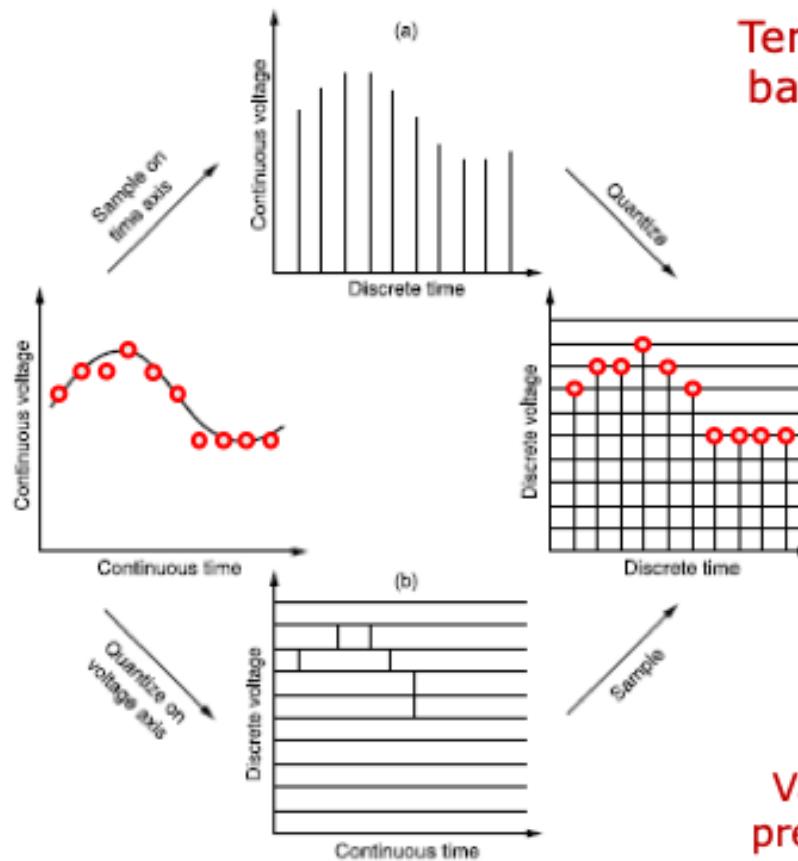
---

# QUANTIZZAZIONE

# Conversione da analogico a digitale



**DOMINIO ANALOGICO**  
variazione di un valore  
continuo con precisione  
infinita



Tempo discreto  
banda limitata

**DOMINIO DIGITALE**  
valori discreti  
(numerabili) con  
precisione finita

Valori discreti  
precisione finita

Esempio CDROM.

Frequenza di campionamento: 44'100 Hz => preserva fino a  $\sim 22$  kHz.

Risoluzione del quantizzatore: 16 bit => 65'536 valori

# Campionamento

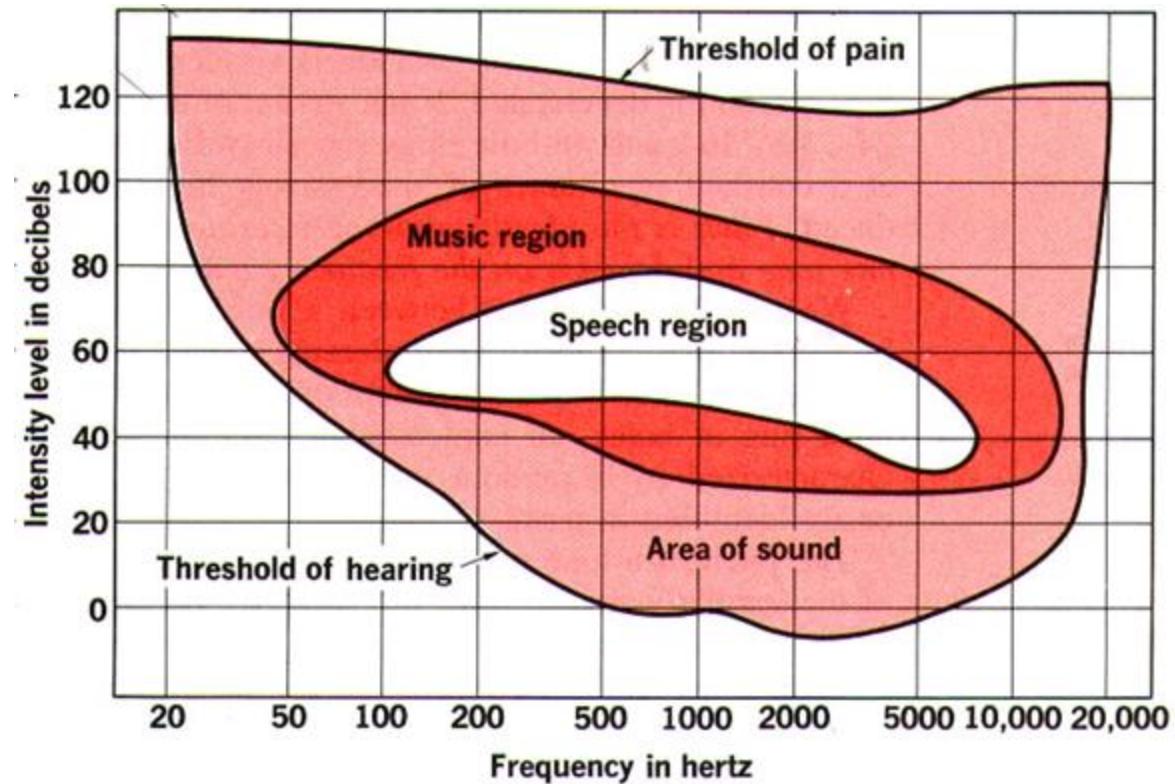
---

- Campionamento + filtraggio (perdita di banda - rimozione frequenze)
  - ✓ Definizione della banda del segnale da preservare e quindi della frequenza di taglio  $F_s \Rightarrow$  banda dell'udibile
  - ✓ Teorema di Shannon / Nyquist: si preservano (idealmente) le frequenze fino a  $F_s/2$
- Quantizzazione (perdita di precisione)
  - ✓ Definizione del rumore di quantizzazione  $\Rightarrow$  soglia di udibilità
  - ✓ Progetto del quantizzatore:
    - N numero di bit per campione,
    - $X_m$  intervallo di quantizzazione,
    - curva caratteristica (e.g. lineare uniforme)

Complesso, ci arriviamo per gradi.

# Campionamento (range di frequenze)

- Audio generale
  - ✓ Tutto quello che l'orecchio può percepire: 20 – 20'000 Hz
- Musica
  - ✓ Intervallo leggermente più ristretto 50 – 16'000 Hz  
ppp~40dB <> ff~100dB
- Voce
  - ✓ Wide band: 80 – 8'000 Hz
  - ✓ Narrow band: 300 - 3'400 Hz
  - ✓ Sussurrato ~30 dB  
Urlato ~90 dB



Differenza tra audio e suono?

# Campionamento + filtraggio

---

- Teorema di Shannon / Nyquist
  - ✓ Data una frequenza di campionamento  $F_s$
  - ✓ Si preservano (idealmente) le frequenze fino a  $F_s/2$
- Frequenza di campionamento
  - ✓ Definito il limite superiore delle frequenze percepibili dall'orecchio  $F_{max}$
  - ✓ Filtraggio: passa basso con frequenza di taglio  $f = F_{max} + \Delta$ 
    - $\Delta$  perché il filtro non è ideale
  - ✓ Campionamento:  $F_s = 2 \times f$
- Perdita di frequenze (alte) => ascolto
  - ✓ Si perde *brillantezza, respiro, spazialità* del suono
  - ✓ Il suono appare *ovattato*

**DEMO – REAPER**  
demo-band-filter-bit-reduction.rpp

# Quantizzazione

---

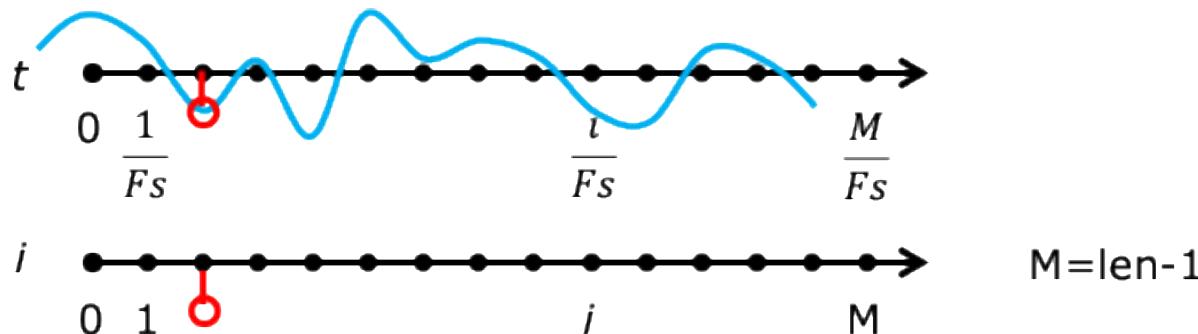
- Progetto del quantizzatore ( $N$ ,  $X_m$ , curva caratteristica)
  - ✓ Rappresentazione in aritmetica intera
  - ✓ Dato un numero di bit  $N$  si hanno solo  $2^N$  valori possibili per rappresentare il valore dei campioni
  - ✓ Definizione dell'intervallo di quantizzazione  $[-X_m, +X_m]$
  - ✓ Definizione della funzione di quantizzazione (curva caratteristica)
    - Lineare: intervallo  $2X_m$  diviso in  $2^N$  parti uguali
- Perdita di precisione => ascolto
  - ✓ Errore di quantizzazione:  $|e| \leq 2 X_m / 2^N$
  - ✓ Al suono originale si aggiunge RUMORE (BIANCO, se Q. bene)

**DEMO – REAPER**  
demo-band-filter-bit-reduction.rpp

---

# Campionamento

- Il campionamento ad una determinata frequenza ( $F_s$ : sampling frequency) fa sì che ogni campione sia associato ad un preciso istante (detto istante di campionamento,  $T_{s_i}$ : sampling time)
- La distanza (tempo) tra un campione e il successivo  $\Delta T_s$  è data dall'inverso di  $F_s$ :  $1/F_s$
- Numerando i campioni con  $i = 0..M$ ,  $T_{s_i}$  è dato da  $i \cdot \Delta T_s$ 
  - ✓ La durata in secondi è quindi data da  $M/F_s$

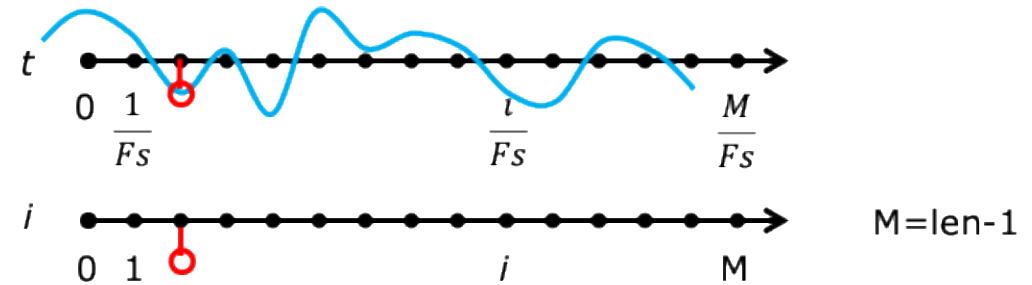


Notazione:  $F_s$  o  $sr$  (sempre sample rate è)

# Campionamento - esempio

- Sia dato un segnale di durata 100 ms con  $F_s = 48 \text{ kHz}$
- Calcolare il numero di campioni  $M$
- Calcolare  $\Delta T_s$
- Calcolare i valori  $i_1$  e  $i_2$  corrispondenti ai campioni (prossimi) a  $t_{i_1}=34\text{ms}$  e  $t_{i_2}=65\text{ms}$

- Il campionamento ad una determinata frequenza ( $F_s$ : sampling frequency) fa sì che ogni campione sia associato ad un preciso istante (detto istante di campionamento,  $T_{s_i}$ : sampling time)
- La distanza (tempo) tra un campione e il successivo  $\Delta t$  è data dall'inverso di  $F_s$ :  $1/F_s$
- Numerando i campioni con  $i = 0..M$ ,  $T_{s_i}$  è dato da  $i \cdot \Delta t$ 
  - ✓ La durata in secondi è quindi data da  $M/F_s$



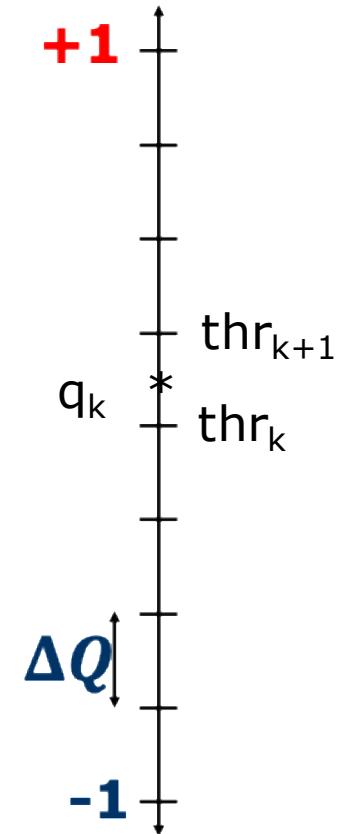
# Campionamento - esempio

---

- Sia dato un segnale di durata 100 ms con  $F_s = 48 \text{ kHz}$
- Calcolare il numero di campioni  $M$ 
  - ✓  $M = \text{round}( (100 / 1000) * 48'000 )$
- Calcolare  $\Delta T_s$ 
  - ✓  $\Delta T_s = 1/F_s$ ,  $\Delta T_s (\text{ms}) = 1/48'000 * 1000 = 1/48 = 0,02 \text{ ms (circa)}$
- Calcolare i valori  $i_1$  e  $i_2$  corrispondenti ai campioni (prossimi) a  $t_{i_1} = 34\text{ms}$  e  $t_{i_2} = 65\text{ms}$ 
  - ✓  $i_1 = \text{round}(0.034 / \Delta T_s) = 34 * 48 = 1'632$
  - ✓  $i_2 = \text{round}(0.065 / \Delta T_s) = 65 * 48 = 3'120$

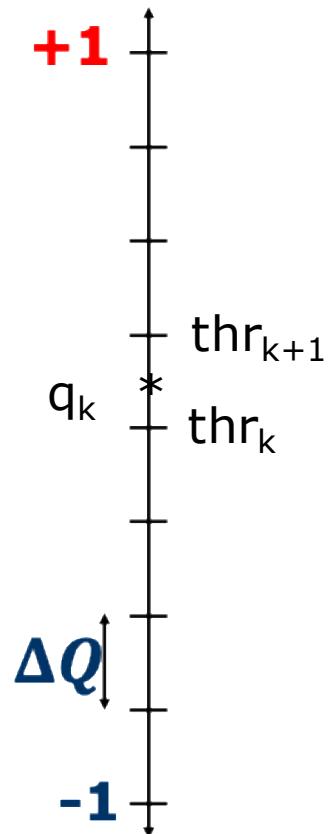
# Quantizzatore - progetto

- Numero di valori/livelli  $2^N$  (n. di bit N),
- Intervallo di quantizzazione / range  $[-X_m, +X_m]$
- Funzione di trasferimento / curva caratteristica
  - ✓ Lineare uniforme: intervallo  $2X_m$  diviso in  $2^N$  parti uguali
- Passo di quantizzazione:  $Q = 2X_m / 2^N$
- Soglie del quantizzatore:  $d_k = -X_m + k * Q$
- Livelli (valori) del quantizzatore:  $q_k = 0.5 * (d_k + d_{k+1})$



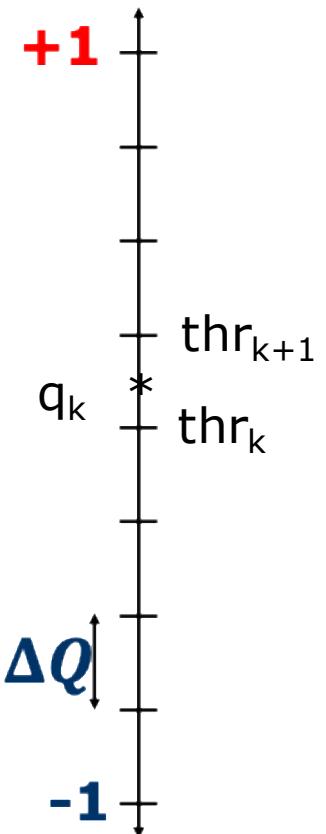
# Quantizzatore – esempio didattico

- Sia dato un quantizzatore lineare con  $X_m = 1$  e  $N = 3$  bit
- Calcolare il passo di quantizzazione
- Calcolare le soglie del quantizzatore
- Calcolare i livelli del quantizzatore
- Calcolare il valore in ingresso 0.01 su che livello in uscita viene quantizzato e quale è l'errore commesso
- Su che livello viene quantizzato il valore +1.2 ?



# Quantizzazione – esempio didattico

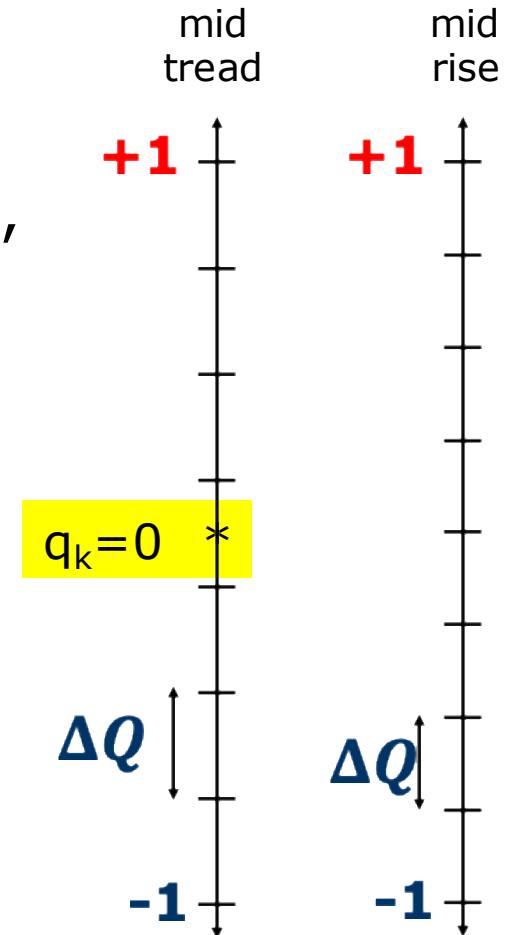
- Sia dato un quantizzatore lineare con  $X_m = 1$  e  $N = 3$  bit
- Calcolare il passo di quantizzazione
  - ✓  $Q = 2 * X_m / 2^N = 2*1/8 = 0.25$
- Calcolare le soglie del quantizzatore
  - ✓  $[ -1, -1+0.25, -1+2*0.25 .. 1]$
- Calcolare i livelli del quantizzatore
  - ✓  $(-1 + (-1+0.25)) / 2 = -0.875 ..$
  - ✓  $[-0.875 \ -0.625 \ -0.375 \ -0.125 \ 0.125 \ 0.375 \ 0.625 \ 0.875]$
- Calcolare il valore in ingresso 0.01 su che livello in uscita viene quantizzato e quale è l'errore commesso
  - ✓  $0.01 \Rightarrow 0.125$ , errore = 0.115



# Quantizzazione mid-tread

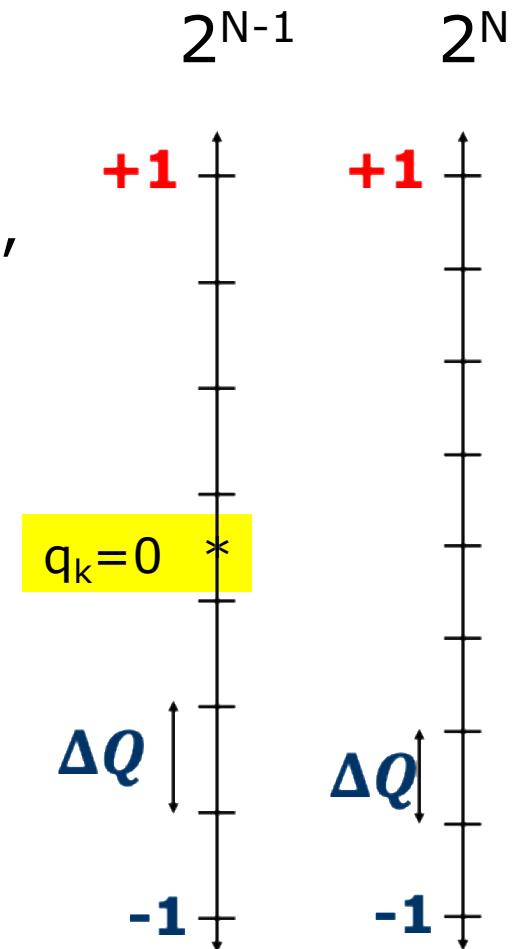
- In ambito audio è un problema non avere la rappresentazione esatta dello zero
  - ✓ Il silenzio non può essere rappresentato correttamente, diventa rumore (a bassa intensità)
- Si può modificare il quantizzatore per avere la rappresentazione dello zero

Trova le differenze



# Quantizzazione mid-tread

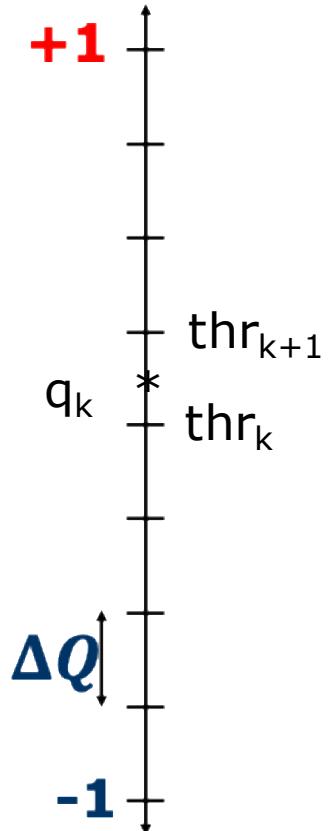
- In ambito audio è un problema non avere la rappresentazione esatta dello zero
  - ✓ Il silenzio non può essere rappresentato correttamente, diventa rumore (a bassa intensità)
- Si può modificare il quantizzatore per avere la rappresentazione dello zero
  - ✓ E' sufficiente "rinunciare" ad un livello/intervallo da  $2^N$  a  $2^{(N-1)}$



# Quantizzatore uniforme mid tread

```
def quantize_uniform_tread(x, N=8):
    qmin, qmax, qlevel = -1, +1, 2**N-1 # defaults
    qstep = (qmax-qmin) / qlevel
    xnorm = (x-qmin) * qlevel / (qmax-qmin)
    xnorm[xnorm > qlevel] = qlevel
    xnorm[xnorm < 0] = 0
    xnorm_quant = np.floor(xnorm)
    xquant = xnorm_quant * (qmax-qmin) / qlevel
    xquant = xquant + qmin + qstep/2
    return xquant

quant_min, quant_max, n = -1, 1, 1000
N = 3
x = np.linspace(quant_min, quant_max, n)
xq = quantize_uniform_tread(x, N)
```



Reference: [https://www.audiolabs-erlangen.de/resources/MIR/FMP/C2/C2S2\\_DigitalSignalQuantization.html](https://www.audiolabs-erlangen.de/resources/MIR/FMP/C2/C2S2_DigitalSignalQuantization.html)

---

# AUDIO DIGITALE CON PYTHON

# Riferimenti – Python per audio digitale

## ■ Introduzione a Python

<https://www.audiolabs-erlangen.de/resources/MIR/FMP/B/B.html>

AUTONOMO

- ✓ Python basics: data types, control structures, functions, numpy
- ✓ Multimedia: audio objects (audio files, audio signals)
- ✓ Python visualization (matplotlib): basic, waveform
- ✓ Python audio: librosa, pysoundfile

## ■ Python Notebooks for Fundamentals of Music Processing

<https://www.audiolabs-erlangen.de/resources/MIR/FMP/C0/C0.html>

## ■ Librosa: python package for music and sound analysis

<https://librosa.org/doc/latest/index.html>

Slides: python

# Caricare e decodificare file audio

- Per essere processati i file audio devono essere decodificati
- E' necessario utilizzare delle librerie/funzioni che ritornano
  - ✓ Dati: una matrice di M canali x N valori float nell'intervallo [-1,+1)
  - ✓ Metadati: la frequenza di campionamento (Fs), ecc.
  - ✓ Attenzione: non tutte le librerie supportano gli stessi e/o tutti i codec



```
import numpy as np
import librosa
filename = 'potter.wav'
w, Fs = librosa.load(filename, sr=None)
```

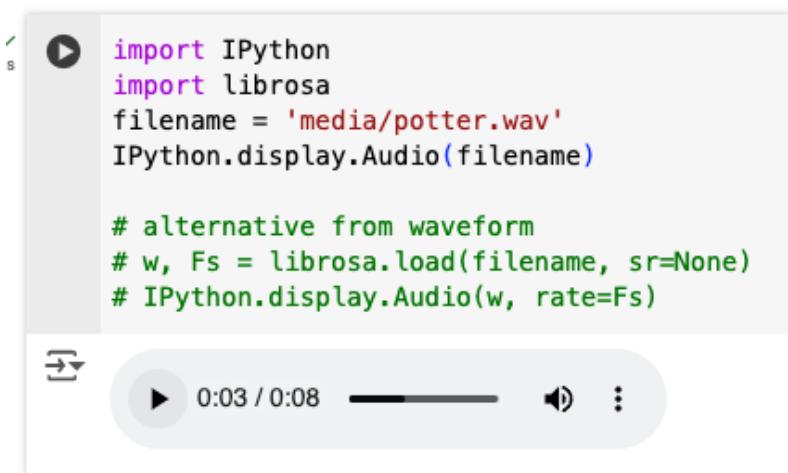
```
ffprobe -i potter.wav
Input #0, wav, from 'potter.wav':
Duration: 00:00:08.80, bitrate: 706 kb/s
Stream #0:0: Audio: pcm_s16le ([1][0][0][0] / 0x0001), 44100 Hz, 1 channels, s16, 705 kb/s
```

Evita ricampionamento  
default a 22 kHz.

# Riproduzione audio

## ■ Utilizzare IPython.display.Audio

- ✓ IPython.display.Audio(<filename>)
- ✓ IPython.display.Audio(<samples>, rate=<rate>)



The image shows a Jupyter Notebook cell with the following code:

```
import IPython
import librosa
filename = 'media/potter.wav'
IPython.display.Audio(filename)

# alternative from waveform
# w, Fs = librosa.load(filename, sr=None)
# IPython.display.Audio(w, rate=Fs)
```

Below the code, there is a media player interface with a play button, a progress bar showing 0:03 / 0:08, and other control buttons.

FMP – Python Multimedia: [https://www.audiolabs-erlangen.de/resources/MIR/FMP/B/B\\_Multimedia.html](https://www.audiolabs-erlangen.de/resources/MIR/FMP/B/B_Multimedia.html)

# Durata e istanti di campionamento

---

- Alcune informazioni non sono restituite e devono essere calcolate
- Durata
  - ✓ E' il rapporto tra il numero di campioni e la frequenza di campionamento  $N/F_s$  (in secondi)
- Istanti di campionamento
  - ✓ La distanza tra un campione e il successivo  $\Delta t$  è data dall'inverso della frequenza di campionamento  $1/F_s$  (in secondi)
  - ✓ Gli istanti di campionamento corrispondono all'indice del campione nel vettore audio  $0, 1, \dots N * 1/F_s$

```
w, Fs = librosa.load(filename, sr=None)
t = np.arange(w.shape[0]) / Fs
```

*ndarray.shape: the number of elements in each dimension*

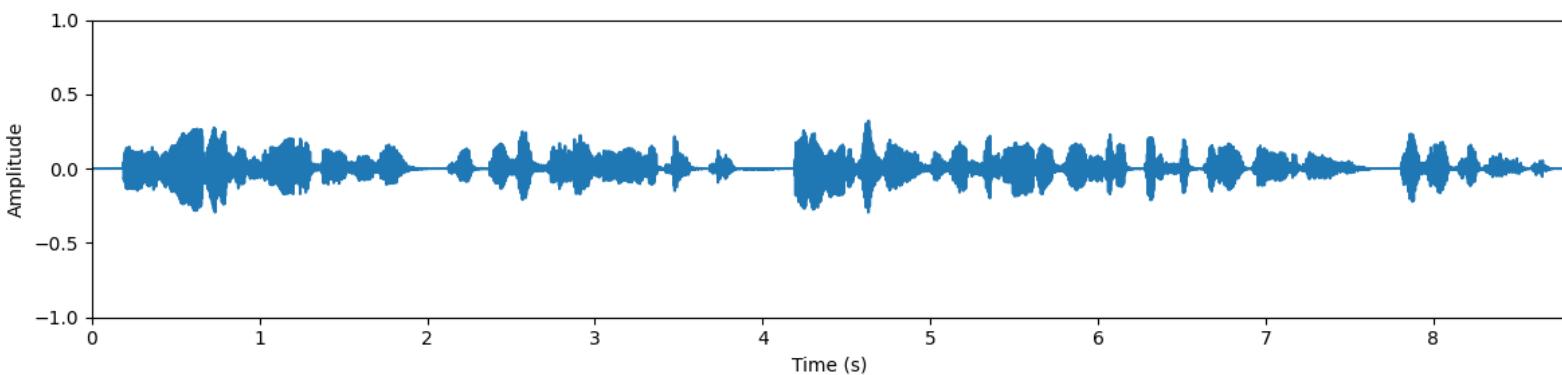
<https://www.geeksforgeeks.org/numpy-ndarray/>

Array in Numpy is a table of elements (usually numbers), all of the same type, indexed by a tuple of positive integers. In Numpy, number of dimensions of the array is called rank of the array. A tuple of integers giving the size of the array along each dimension is known as shape of the array. An array class in Numpy is called as **ndarray**. Elements in Numpy arrays are accessed by using square brackets and can be initialized by using nested Python Lists.

# Rappresentazione nel tempo

- Per rappresentare la forma d'onda nel tempo occorre
  - ✓ Il vettore **t** con gli istanti di campionamento
  - ✓ Il vettore **w** con i valori dei campioni, le "ampiezze"
  - ✓ E poi è bene aggiungere le etichette e i limiti degli assi

```
from matplotlib import pyplot as plt
fig, ax = plt.subplots(figsize=(12,3))
ax.plot(t, w, '-')
ax.set_xlabel('Time (seconds)')
ax.set_ylabel('Amplitude')
ax.set_xlim([t[0], t[-1]])
ax.set_ylim([-1, 1])
ax.grid()
fig.tight_layout()
```



FMP – Python Visualization: [https://www.audiolabs-erlangen.de/resources/MIR/FMP/B/B\\_PythonVisualization.html](https://www.audiolabs-erlangen.de/resources/MIR/FMP/B/B_PythonVisualization.html)

# Scrittura / codifica di file audio

---

- E' necessario utilizzare librerie con il supporto al formato di codifica
  - ✓ Librosa all'interno si appoggia su soundfile (PySoundFile) e audioread
  - ✗ ~~Si può usare librosa.output.write\_wav(path, y, sr [, norm])~~
- PySoundFile (read and write API)
  - ✓ <https://pysoundfile.readthedocs.io/en/latest/>
  - ✓ soundfile.write
    - file (string): file pathname
    - data (float/int): samples (N samples x M channels) *transposed!*
    - samplerate (int): sample rate [output]
    - subtype (str, optional): audio sample format [output] *encoding!*
    - format (str, optional): file container format [output]  
(can be otherwise inferred by the file extension)

FMP – Python Audio: [https://www.audiolabs-erlangen.de/resources/MIR/FMP/B/B\\_PythonAudio.html](https://www.audiolabs-erlangen.de/resources/MIR/FMP/B/B_PythonAudio.html)

---

# PySoundFile

```
# PySoundFile

# available subtypes for output audio coding
print("Codecs")
pprint(sf.available_subtypes())

# available formats for output audio container
print("Containers")
pprint(sf.available_formats())
```

Codecs

```
{'ALAC_16': '16 bit ALAC',
 'ALAC_20': '20 bit ALAC',
 'ALAC_24': '24 bit ALAC',
 'ALAC_32': '32 bit ALAC',
 'ALAW': 'A-Law',
 'DOUBLE': '64 bit float',
 'DPCM_16': '16 bit DPCM',
 'DPCM_8': '8 bit DPCM',
 'DWVW_12': '12 bit DWVW',
 'DWVW_16': '16 bit DWVW',
 'DWVW_24': '24 bit DWVW',
 'FLOAT': '32 bit float',
 'G721_32': '32kbs G721 ADPCM',
 'G723_24': '24kbs G723 ADPCM',
 'G723_40': '40kbs G723 ADPCM',
 'GSM610': 'GSM 6.10',
 'IMA_ADPCM': 'IMA ADPCM',
 'MPEG_LAYER_I': 'MPEG Layer I',
 'MPEG_LAYER_II': 'MPEG Layer II',
 'MPEG_LAYER_III': 'MPEG Layer III',
 'MS_ADPCM': 'Microsoft ADPCM',
 'NMS_ADPCM_16': '16kbs NMS ADPCM',
 'NMS_ADPCM_24': '24kbs NMS ADPCM',
 'NMS_ADPCM_32': '32kbs NMS ADPCM',
 'OPUS': 'Opus',
 'PCM_16': 'Signed 16 bit PCM',
 'PCM_24': 'Signed 24 bit PCM',
 'PCM_32': 'Signed 32 bit PCM',
 'PCM_S8': 'Signed 8 bit PCM',
 'PCM_U8': 'Unsigned 8 bit PCM',
 'ULAW': 'U-Law',
 'VORBIS': 'Vorbis',
 'VOX_ADPCM': 'VOX ADPCM'}
```

Containers

```
{'AIFF': 'AIFF (Apple/SGI)',
 'AU': 'AU (Sun/NeXT)',
 'AVR': 'AVR (Audio Visual Research)',
 'CAF': 'CAF (Apple Core Audio File)',
 'FLAC': 'FLAC (Free Lossless Audio Codec)',
 'HTK': 'HTK (HMM Tool Kit)',
 'IRCAM': 'SF (Berkeley/IRCAM/CARL)',
 'MAT4': 'MAT4 (GNU Octave 2.0 / Matlab 4.2)',
 'MAT5': 'MATS (GNU Octave 2.1 / Matlab 5.0)',
 'MP3': 'MPEG-1/2 Audio',
 'MPC2K': 'MPC (Akai MPC 2k)',
 'NIST': 'WAV (NIST Sphere)',
 'OGG': 'OGG (OGG Container format)',
 'PAF': 'PAF (Ensoniq PARIS)',
 'PVF': 'PVF (Portable Voice Format)',
 'RAW': 'RAW (header-less)',
 'RF64': 'RF64 (RIFF 64)',
 'SD2': 'SD2 (Sound Designer II)',
 'SDS': 'SDS (Midi Sample Dump Standard)',
 'SVX': 'IFF (Amiga IFF/SVX8/SV16)',
 'VOC': 'VOC (Creative Labs)',
 'W64': 'W64 (SoundFoundry WAVE 64)',
 'WAV': 'WAV (Microsoft)',
 'WAVEX': 'WAVEX (Microsoft)',
 'WVE': 'WVE (Psion Series 3)',
 'XI': 'XI (FastTracker 2)'}
```

---

## FORMATI AUDIO

# Introduzione - distinzione

---

- Un file audio è composto da due aspetti distinti
- 1. Formato della struttura del file (contenitore)
  - ✓ Come sono organizzate le informazioni, con intestazioni (header), metadati e dati multimediali
- 2. Formato di codifica del segnale (dati audio)
  - ✓ Come il segnale audio è rappresentato o compresso dentro il contenitore
- Esempio intuitivo
  - ✓ In un file A/V abbiamo il formato del file (collegato all'estensione dello stesso, e.g. mp4) e due codifiche una per l'audio e una per il video

# Formati del contenitore

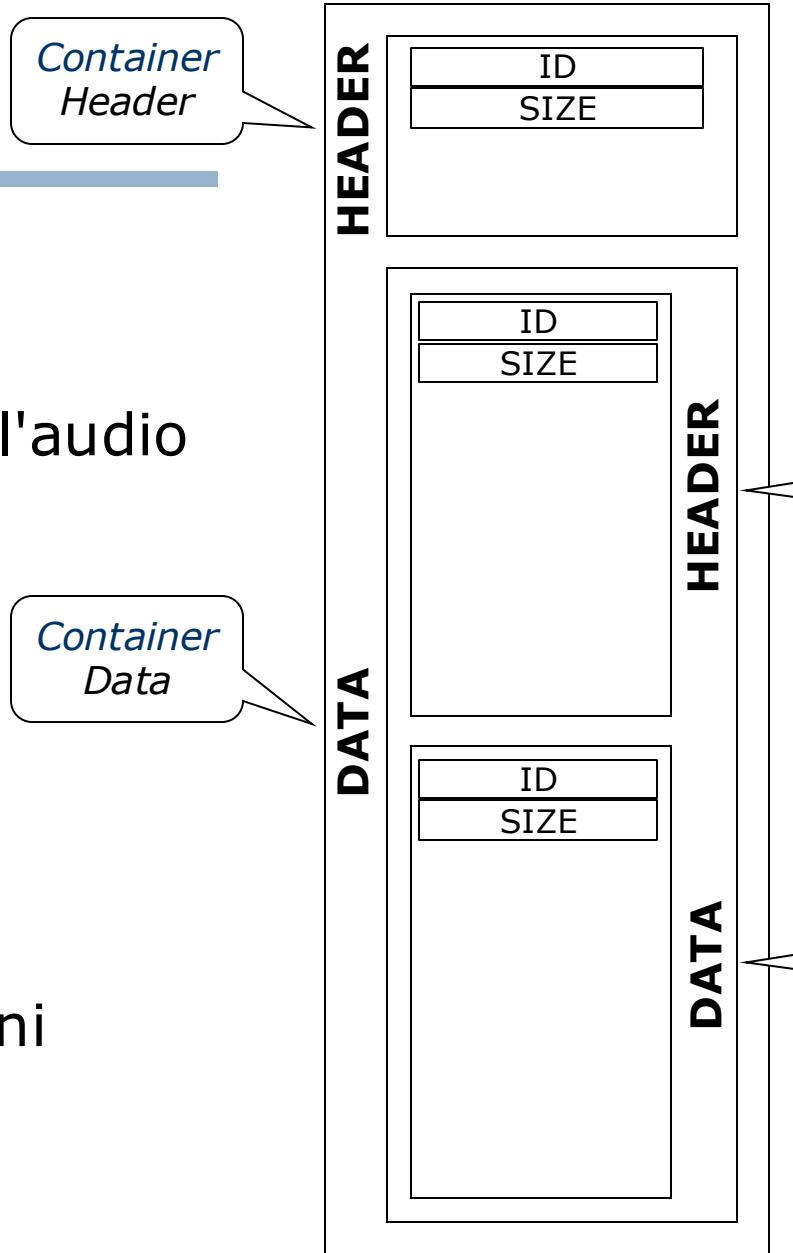
---

- Tra i formati del **contenitore** più diffusi abbiamo
- Audio non compresso:
  - ✓ WAV (.wav): standard Microsoft/IBM 1991, AIFF (.aiff): analogo Apple
  - ✓ Contiene praticamente sempre audio PCM (q. lineare uniforme)
- Audio compresso percettivo (lossy):
  - ✓ .MP3, .AAC / .M4A: famiglia standard MPEG / ISO - brevetti
  - ✓ .OGG / .OPUS: famiglia Xiph (open source)
- Audio compresso (lossless):
  - ✓ .FLAC: famiglia Xiph (open source)
  - ✓ .ALAC/.M4A: famiglia MPEG / APPLE

# Esempio WAV

- Il file è composto da metadati (header) e dati
  - ✓ L'header identifica il container WAVE
  - ✓ La parte dati contiene la rappresentazione dell'audio
- La rappresentazione dell'audio è a sua volta strutturata in header e dati
  - ✓ L'header descrive il parametri di codifica
    - Frequenza di campionamento
    - Numero di canali
    - Risoluzione (# bit) dei campioni
  - ✓ La parte dati contiene i valori PCM dei campioni

N.B. ogni chunk è definito da un identificativo  
(4 caratteri ASCII, "fmt " / "data") e una dimensione



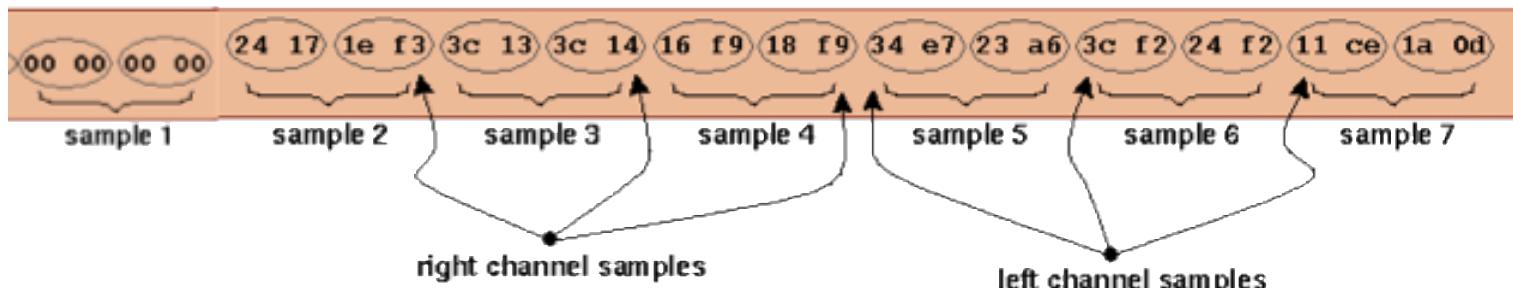
# Esempio WAV

## ■ Formato parte dati audio PCM

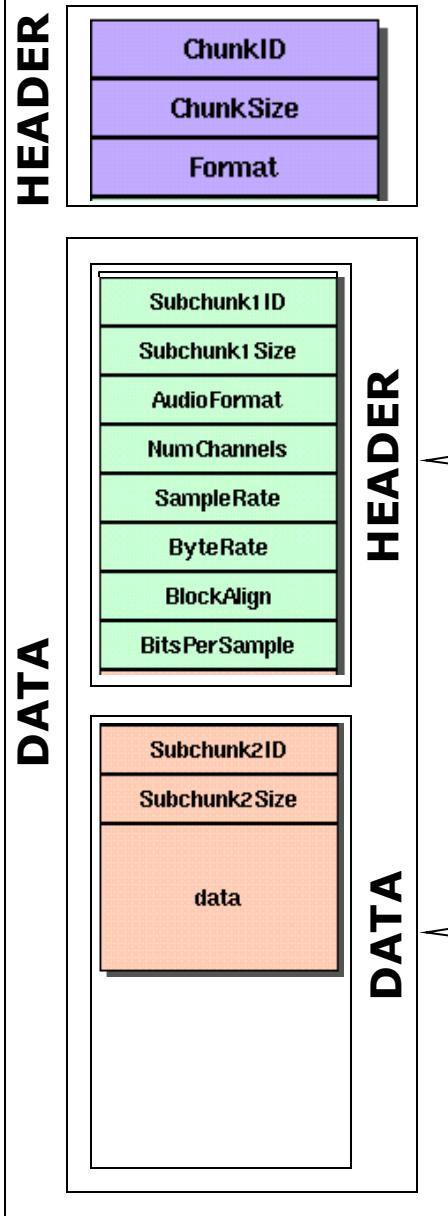
- ✓ 4 byte per l'ID: "data" - 4 byte per la dimensione (size)
- ✓ Dati grezzi: mono o multicanale
  - Campione su 16 bit, 2 byte (HEX): 24 17

## ■ Rappresentazione multicanale: INTERLACCIATA

- ✓ I valori dei campioni sono "raggruppati" per istante di campionamento



*Container Header*



# Codifica del \_segnale\_

---

- Distinguiamo due "famiglie"
- Codifica **campione** per campione
  - ✓ Viene codificata la forma d'onda  
(ogni campione è codificato singolarmente)
  - ✓ L'esempio più esemplare è il PCM (no-compressione)
  - ✓ Utilizzata storicamente per la voce in ambito telefonico
    - PCM lineare, PCM u-law/a-law, PCM adattativo differenziale, ecc.
- Codifica per **frame** (gruppi di campioni contigui nel tempo)
  - ✓ Frame dell'ordine dei 20 ms
  - ✓ Viene codificato lo spettro del frame ("transform" coders)
  - ✓ Largamente utilizzato nei codificatori moderni sia audio sia voce

# Formati di codifica del \_segnale\_

---

- PCM (Pulse Code Modulation)
  - ✓ Codifica per campione, quantizzatore lineare uniforme
  - ✓ Nessuna compressione, elevata qualità, dimensioni elevate
- MP3 (MPEG-Layer III) – AAC (MPEG-4 P.12)
  - ✓ Codifica a frame percettiva: l'informazione in frequenza è analizzata con un modello psicoacustico, l'errore di codifica è mantenuto sotto la curva di mascheramento del segnale (non udibile) + entropica
  - ✓ Compressione percettiva  $\sim 10:1$
  - ✓ Perdita di qualità teoricamente irrilevante
- FLAC (Free Lossless Audio Codec) - MPEG-4 ALS (Audio Lossless)
  - ✓ Codifica a frame predizione + entropica  $\sim 2:1$
  - ✓ Nessuna perdita di qualità

# Compressione percettiva lossy

- Per compressione lossy (percettiva) si intende che l'audio riprodotto (decodificato) SUONA COME l'originale, ma non è identico bit per bit all'originale
    - ✓ Se si usa un numero sufficiente di bit si perde sì dell'informazione, ma solo quella che è irrilevante per il nostro sistema uditivo
    - ✓ Codifica "trasparente" AAC stereo 128-192 kb/s vs. 1411 kb/s WAV 16 bit 44.1 kHz
  - Formati
    - ✓ MP3 (1991), AAC (1997): famiglia MPEG / ISO
    - ✓ AC3 (1991): Dolby – famoso perché usato nei DVD
      - Adesso abbiamo ATMOS, AC-4 (cinema, broadcast, streaming, immersivo)
    - ✓ VORBIS (2000) - Xiph, OPUS (2012) - IETF: formati aperti
- LA QUALITA' E'  
FUNZIONE  
del BITRATE e  
del CODEC

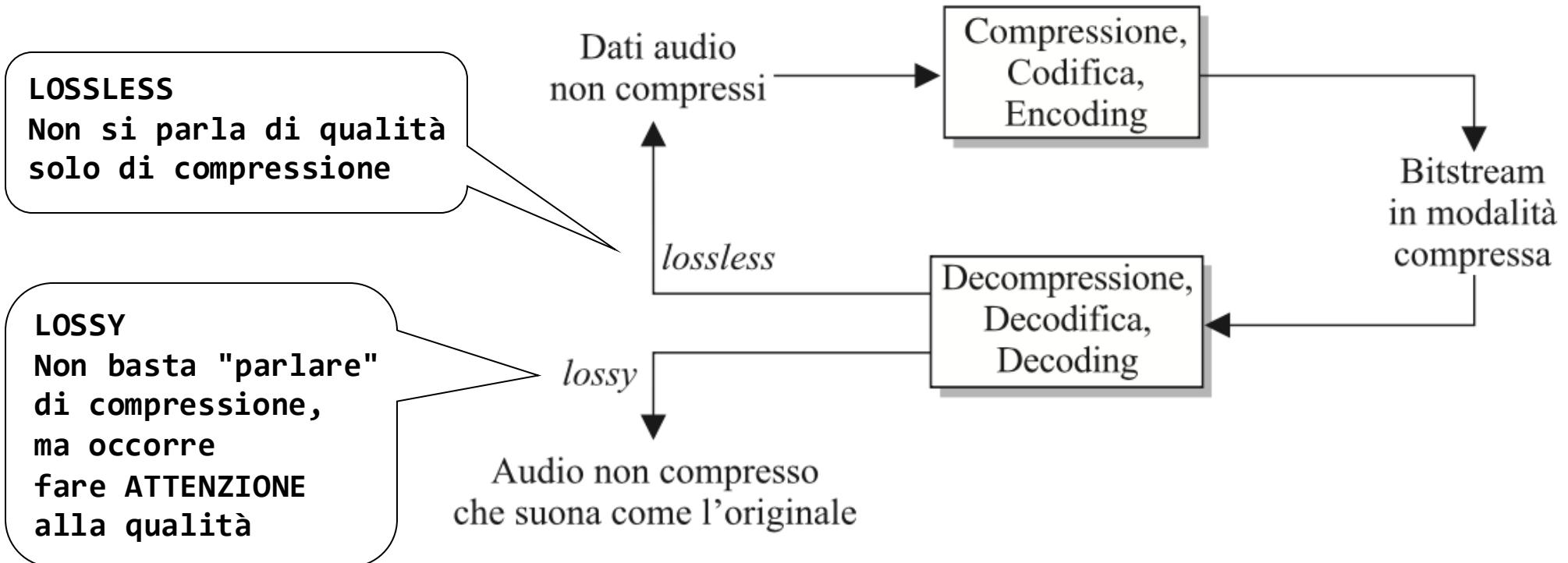
# Compressione lossless

---

- Per compressione lossless si intende che l'audio riprodotto (decodificato) è effettivamente IDENTICO (bit-perfect) all'originale
- Compressione entropica
  - ✓ Predizione: stima dei campioni futuri in funzione dei precedenti
    - Codificare l'errore di predizione richiede meno bit del valore intero
  - ✓ Codifica entropica: compressione tipo "ZIP"
  - ✓ Rapporto di compressione circa 2:1 rispetto a WAV
- Formati
  - ✓ FLAC "Free Lossless Audio Codec" – Xiph (2001)
  - ✓ ALAC "Apple Lossless Audio Codec" – Apple (2004)
  - ✓ MPEG-4 ALS "Audio Lossless Coding" – MPEG (2006)

# Lossless vs Lossy

- La qualità di codifica lossy dipende dalla "bontà" del codificatore
- La compressione di codifica lossless dipende dalla "bontà" del codificatore



# FFMPEG

---

- Il coltellino svizzero della codifica multimediale (dal 2000)
  - ✓ Open source, cross platform, command line
  - ✓ Basato su librerie libav\*
  - ✓ Supporta molti formati file e codifiche audio
- Codificatori / Decodificatori
  - ✓ Per i decodificatori non c'è particolare problema
  - ✓ Idem per i codificatori open source
  - ✓ CAVEAT: codificatori percettivi (lossy)
    - Non tutti i codec sono uguali la qualità a parità di bitrate variano
    - Esempio: aac (versione open source), fdk\_aac (versione con licenza sviluppata dal Fraunhofer Institute per Android – di migliore qualità)
      - Molte distribuzioni non includono fdk\_aac in ffmpeg perché non-free

---

## ERRORE DI QUANTIZZAZIONE

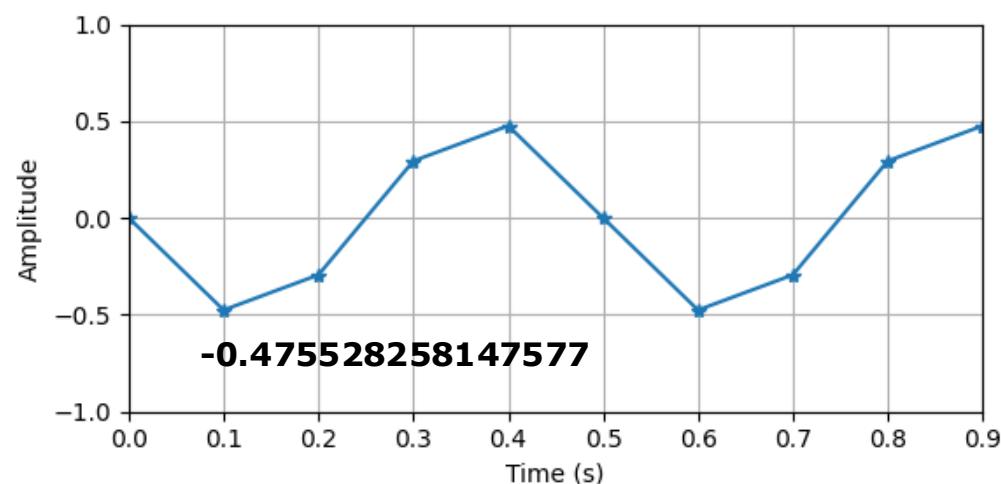
rumore - distorsione

# Quantizzazione

- Una sinusoide sintetizzata al calcolatore ha una elevata precisione di rappresentazione
  - ✓  $w = A \cdot \cos(2\pi \cdot f \cdot t + \varphi)$
  - ✓  $s = A * np.cos(2 * np.pi * f * t + phi)$
- Generalmente sono valori *double / float64*
  - ✓ Su 64 bit (segno, esponente 11 bit, mantissa 52 bit)
  - ✓ Equivalente a 15 cifre *significative*

Cosa significa?

```
f'Sample {np.finfo(s[0]).dtype} precision: '
f'{np.finfo(s[0]).precision} decimal digits'
```



# Esercizio: quantizzazione

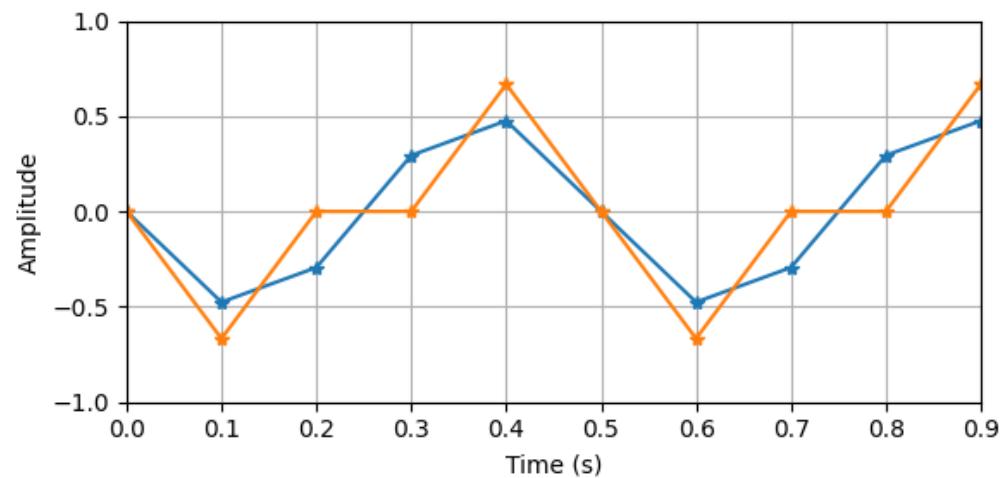
- Utilizzare la funzione *quantize\_uniform* per quantizzare una sinusoide
  - ✓ Utilizzare più valori di ampiezza
  - ✓ e più valori di numero di livelli (qlevel)

```
N, Fs, A, f, phi = 10, 10, 0.5, 2, np.pi/2
t = np.arange(N) / Fs
s = 0.5 * np.cos(2 * np.pi * f * t + phi)

sq = quantize_uniform(s, N=2)

fig, ax = plt.subplots(figsize=(12,3))
plot_vs_time(s,t,fig=(fig,ax))
plot_vs_time(sq,t,fig=(fig,ax))

ax.grid()
```



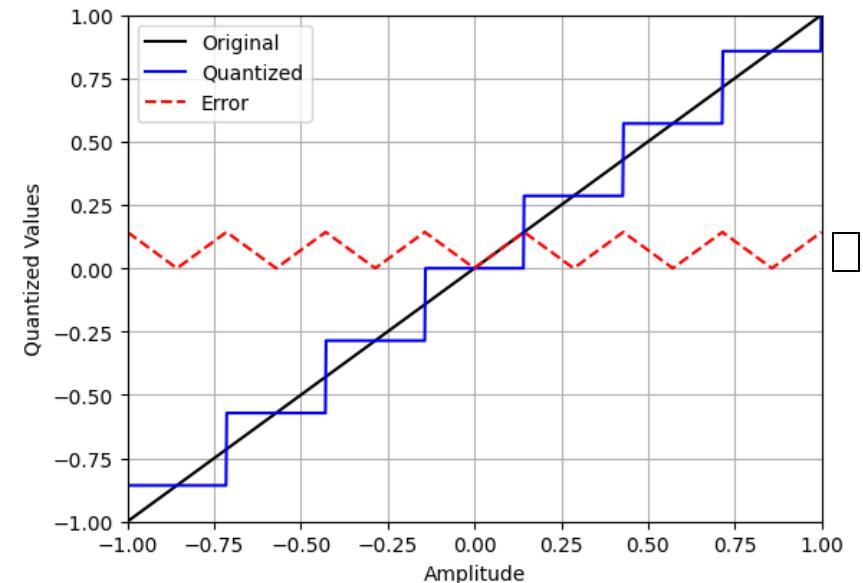
# Errore di quantizzazione

- Il quantizzatore approssima il valore di ciascun campione al valore “più simile” tra quelli (finiti,  $2^N$ ) a sua disposizione
- La differenza tra il valore originale  $x$  e il valore quantizzato  $x_q$  è detta “errore di quantizzazione”  $e$

$$e = \text{abs}(x - x_q)$$

- Per il quantizzatore uniforme che abbiamo definito l’errore, in valore assoluto, è sempre inferiore a metà dell’intervallo di quantizzazione:

$$|e| \leq \Delta/2$$



# Misurare l'errore di quantizzazione

---

- Quale è l'energia media (normalizzata per unità di tempo) dell'errore di quantizzazione? Cioè la potenza?
- La potenza è definita come il valore atteso del quadrato dei campioni  $P_x = E[x^2] = \frac{1}{N} \sum_{i=0}^{N-1} x_i^2$
- Dal momento che il segnale audio ha media nulla l'energia media coincide con la varianza del segnale

$$\sigma_x^2 = \frac{1}{N} \sum_{i=0}^{N-1} (x[n] - \mu)^2 = E[x^2] = P_x$$

- In Python `powX = np.average(x**2)`

# Signal-to-Noise Ratio

---

- La potenza del rumore non è sufficiente a definire la qualità di un quantizzatore
  - ✓ La qualità è funzione sia della potenza del rumore di q. sia della potenza del segnale
    - Es. il rumore di dell'impianto di aerazione in aula non si sente durante la lezione, ma se si fa silenzio si inizia a percepire (e disturbare)
- Occorre calcolare anche la potenza del segnale
- La qualità di un codificatore è calcolata come RAPPORTO con il Signal-to-Noise Ratio

$$SNR = 10 \log_{10} \left( \frac{\sigma_x^2}{\sigma_e^2} \right) dB = 20 \log_{10} \left( \frac{\sigma_x}{\sigma_e} \right) dB$$

N.B. Si utilizza la scala in decibel per:  
praticità di calcolo, perchè la percezione dell'intensità è logaritmica, perchè il range è molto ampio

---

# Signal to Noise Ratio

---

- La qualità del quantizzatore è misurata valutando il rapporto tra
  - ✓ La potenza del segnale  $x$  e la potenza dell'errore di quantizzazione  $e$
- Il rapporto è espresso in dB, cioè su scala logaritmica, e viene chiamato **SNR: signal to noise ratio**
  - ✓ N.B. Non serve dividere per il numero di campioni perché è lo stesso numero

$$SNR = 10 \log_{10} \left( \frac{\sigma_x^2}{\sigma_e^2} \right) dB$$

```
def SNR(original, quantized):  
    noise = quantized - original  
    powS = np.sum(original**2)  
    powN = np.sum(noise**2)  
    return 10*np.log10(powS/powN)
```

# Esempio: (ri-)quantizzazione ed SNR

---

- Calcolare l'SNR in seguito alla ri-quantizzazione di un file audio
  - ✓ Utilizzare 8 bit e 3 bit

```
filename = 'media/potter.wav'
w, Fs = librosa.load(filename, sr=None)

N = 8
wq = quantize_uniform_tread(w, N)
print(f'N: {N}, SNR: {SNR(w, wq):.2f} dB') # N:8, SNR: 26.87 dB
```

*N.B. ri-quantizzazione  
perchè partiamo da un segnale già quantizzato*

```
def SNR(original, quantized):
    noise = quantized - original
    powS = np.sum(original**2)
    powN = np.sum(noise**2)
    return 10*np.log10(powS/powN)
```

# Prova di ascolto

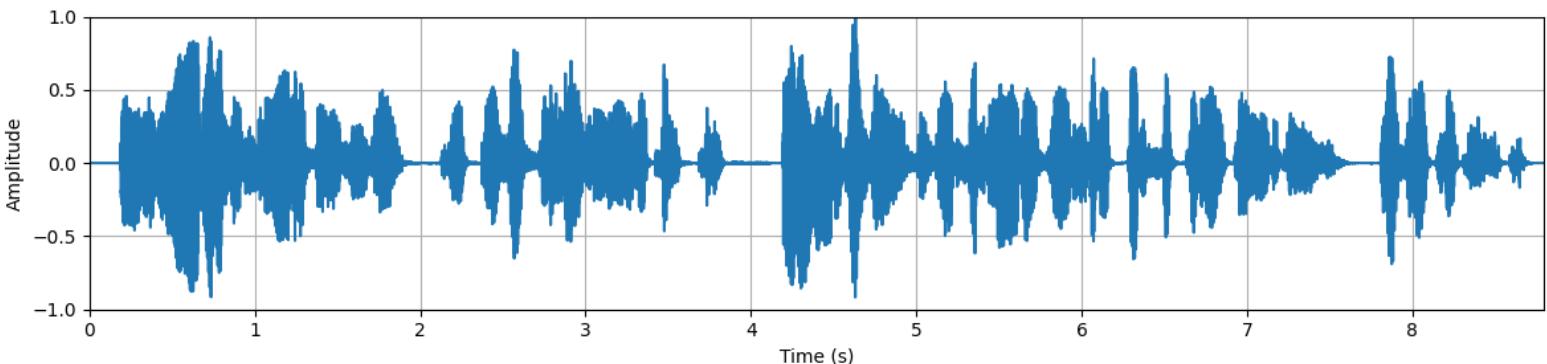
---

- Esercizio:
  - ✓ caricare un file audio e quantizzarlo con più (8) o meno bit (3)
  - ✓ ascoltare il segnale quantizzato e l'errore di quantizzazione
    - n.b. il widget Audio amplifica l'audio prima di riprodurlo
- Soluzione
  - ✓ L'errore dell'audio quantizzato con 3 bit contiene a tratti il segnale audio originale
    - Con pochi bit è poca l'*informazione* che è stata preservata dalla quantizzazione, quella non codificata è *rimasta* nell'errore
- Poll: se dopo la quantizzazione amplifico il segnale, l'SNR aumenta / diminuisce / resta uguale? Perchè?

# Analisi del segnale quantizzato

- Se la dinamica del segnale non è *adattata* alla dinamica del quantizzatore la quantizzazione è sub-ottima
- Proviamo a normalizzare il segnale audio prima di quantizzarlo  
Come cambia l'SNR?
  - Normalizzazione (di picco): scalare il segnale in modo che il campione con valore massimo, in valore assoluto, sia uguale a 1
    - ✓  $x_{\max} = \text{np.max}(\text{np.abs}(x))$
    - ✓  $wn = w/x_{\max}$

36.48  
N:8, SNR: 26.87 dB  
N:3, SNR: 0.85 dB



# Studio della formula dell'SNR

$$SNR = 10 \log_{10} \left( \frac{\sigma_x^2}{\sigma_e^2} \right) dB$$

- Introducendo qualche semplificazione e ponendosi in condizioni ideali con un numero sufficiente di bit di quantizzazione N
  - ✓ Si può assumere che l'errore del quantizzatore sia rumore bianco
- Si può quindi derivare matematicamente che l'SNR equivale a

$$SNR = +6.02 N - 20 \log_{10} \left( \frac{X_m}{\sigma_x} \right) + 4,77$$

- E' quindi funzione del numero dei bit N e del rapporto tra  $X_m$  e  $\sigma_x$ 
  - ✓ Ricordiamo che  $X_m$  è il valore (assoluto) che corrisponde agli estremi dell'intervallo di quantizzazione
  - ✓ Ad esempio  $[-1, +1] \Rightarrow X_m = 1$

Sostituendo

$$SNR = +6.02 N + 20 \log_{10}(\sigma_x) + 4,77$$

# Studio della formula dell'SNR

$$SNR = 10 \log_{10} \left( \frac{\sigma_x^2}{\sigma_e^2} \right) dB$$

- In condizioni ideali (da verificare) assumiamo che

$$SNR \approx +6.02 N + 20 \log_{10}(\sigma_x) + 4,77$$

- Abbiamo che l'SNR aumenta

- ✓ Di 6 dB per ogni bit in più del quantizzatore
  - ✓ Di 6 dB per ogni raddoppio di sigma (deviazione standard)  $20 \log_{10}(2) = 6$ 
    - Cioè se (**prima della quantizzazione !!**) si moltiplica x2 il valore dei campioni  $s^2 = 2*s$
    - Lo abbiamo visto nel caso della "normalizzazione"

- Domanda: fino a quando / quanto vale questo "trucco"?

- ✓ Cioè fino a quanto posso aumentare il livello del mio segnale per migliorare l'SNR di quantizzazione?

# Esercizio

potter.wav

- Dato un segnale audio quale è il valore di  $\sigma$  che massimizza l'SNR?
- Utilizzare Colab e il codice di esempio per calcolare come varia l'SNR in funzione di  $\sigma$  per un audio  $x$ , fissato  $N$  (numero di bit)

```
xA = A * x  
xAq = quantize_uniform_tread(xA, N)  
snr = SNR(xA, xAq) # dB  
rms = RMS(xA)
```

$$\sigma = \sqrt{P_x} = \sqrt{\frac{1}{n} \sum_{i=0}^{n-1} x_i^2} = RMS$$

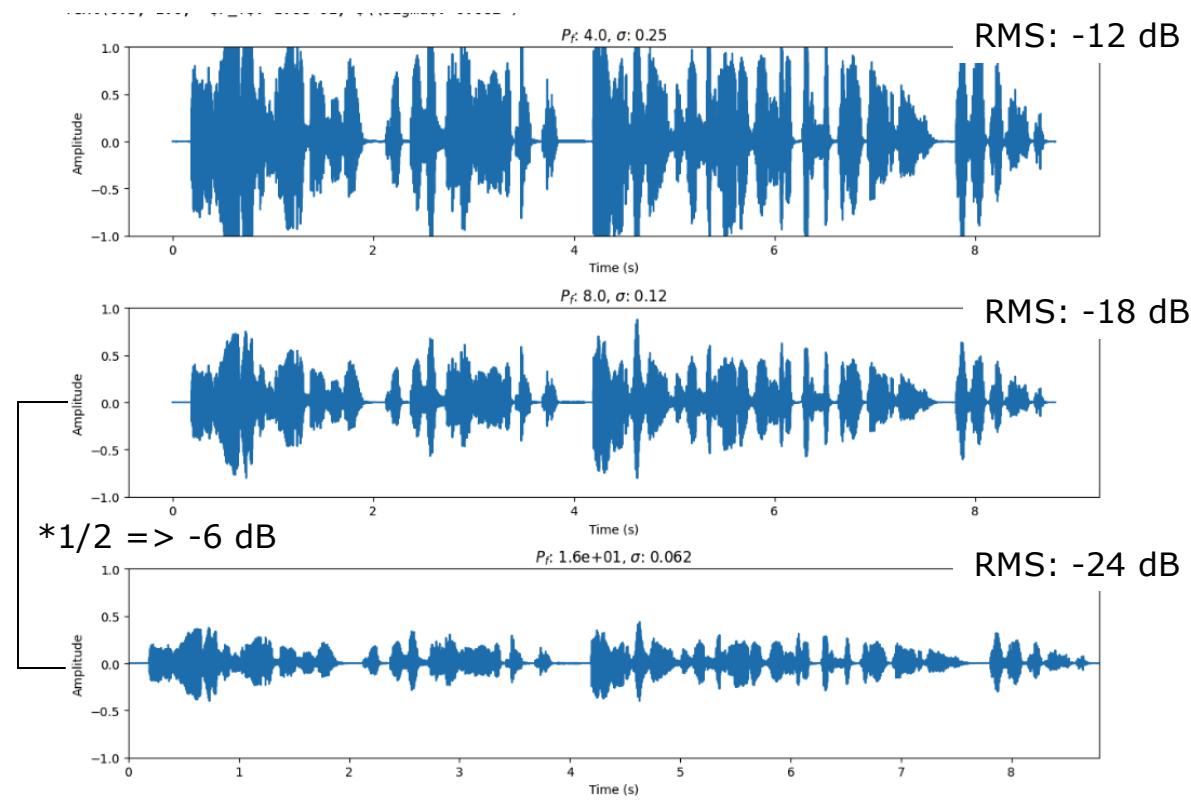
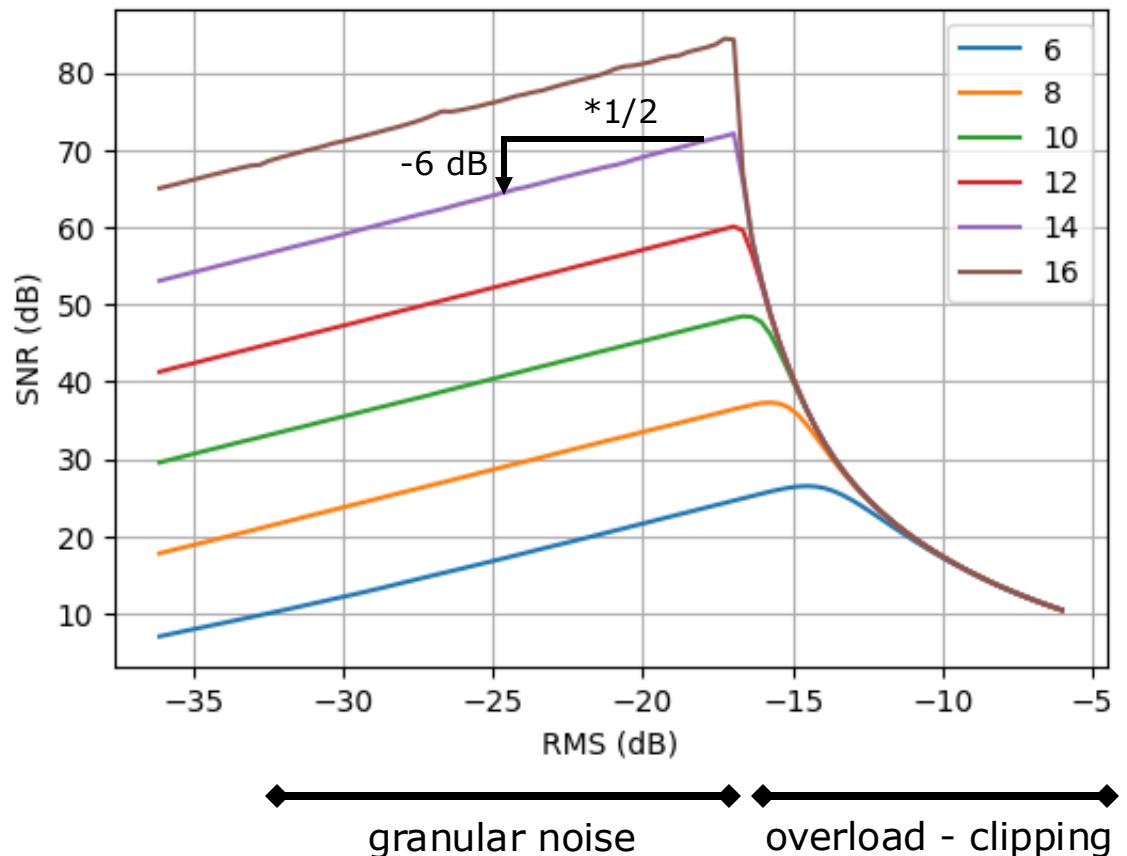
$$RMS(A \cdot x) = |A| \cdot RMS(x)$$

$$X_m=1$$

Gsheet: ead26.quantizer\_snr\_vs\_rms

# Esercizio: soluzione

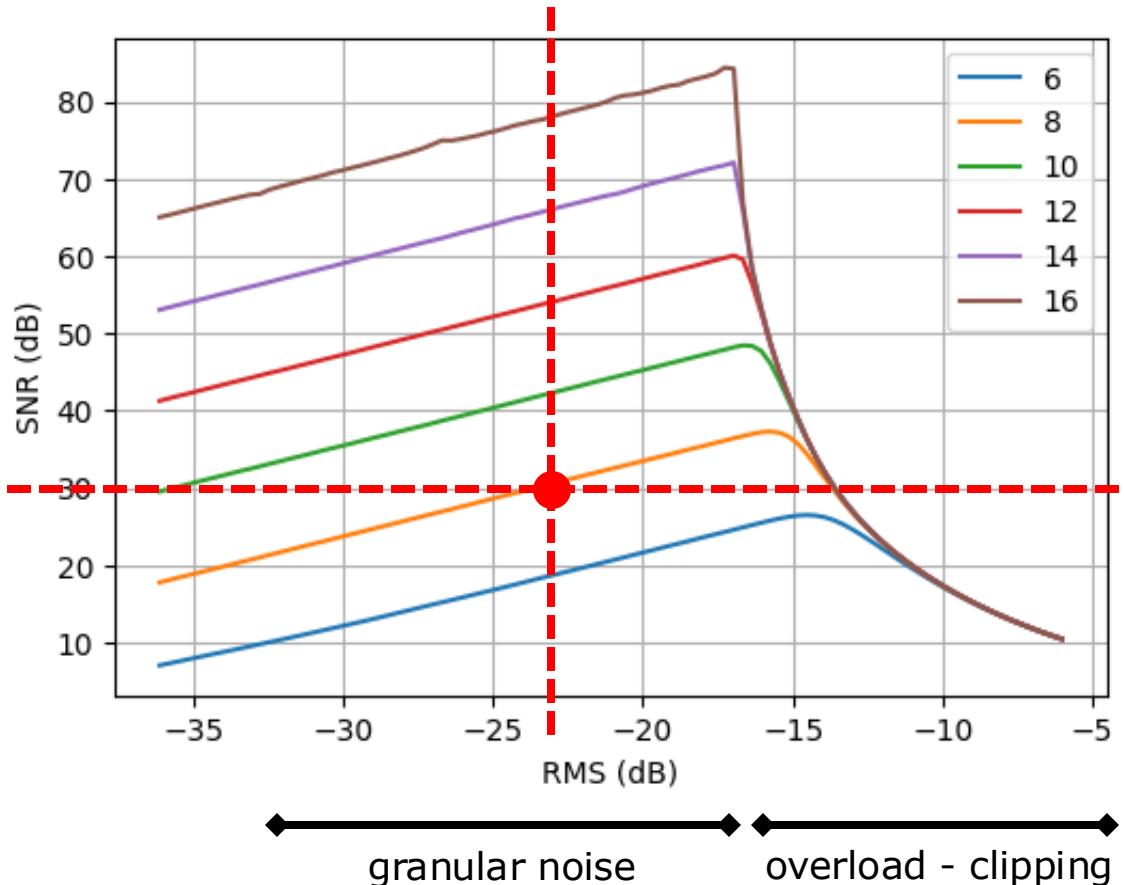
- Come varia l'SNR in funzione di  $\sigma$  per un brano vocale  $x$ , fissato  $N$  (numero di bit)



Gsheet: ead26.quantizer\_snr\_vs\_rms

# Esercizio: ascolto

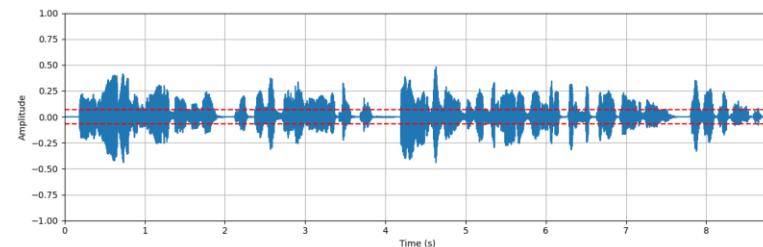
- Come varia l'SNR in funzione di  $\sigma$  per un brano vocale  $x$ , fissato  $N$  (numero di bit)



Riportare nel foglio di calcolo i seguenti valori  
RMS (colonna A): 0.06869259476661682  
SNR (colonna B): 30.32427978515625

Ascoltare l'audio quantizzato.  
Quanto è "buono" un SNR di: 30 dB?

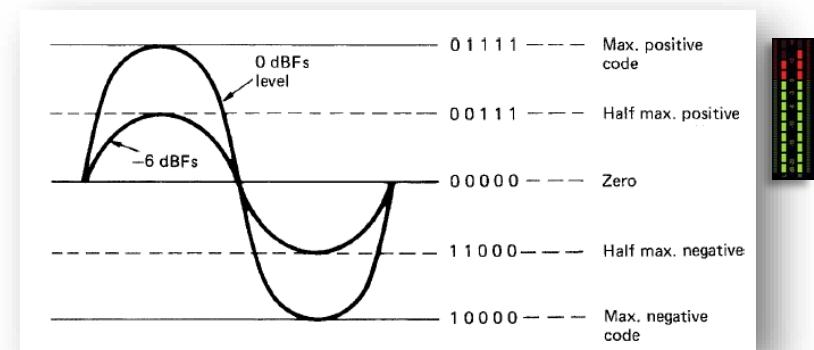
▶ 0:00 / 0:08    ⏸    🔊    ⋮



# Dinamica del segnale vs. SNR

- Abbiamo già visto che se la dinamica del segnale non è adattata a quella del quantizzatore, l'SNR è sub-ottimo.  
Perché?
- Risposta "intuitiva"
  - ✓ Perché si usano meno livelli del quantizzatore e quindi meno bit
  - ✓ Ogni bit che non si sfrutta sono circa 6 dB persi
  - ✓ Ogni dimezzamento della dinamica del segnale causa la perdita di un bit e quindi di 6 dB

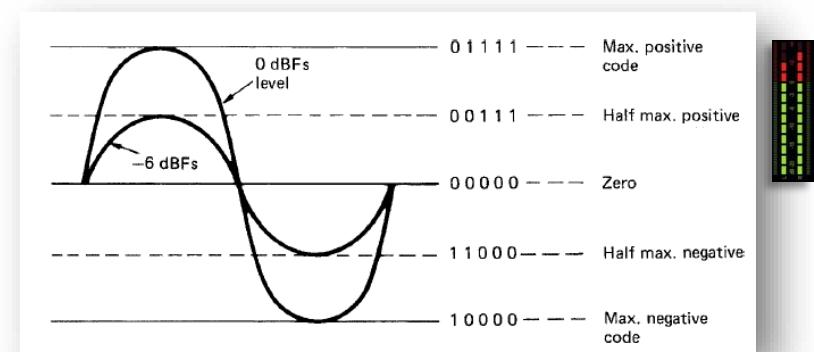
Esempio:  
sinusoide di ampiezza 1 o 0.5



# Dinamica del segnale vs. SNR

- Abbiamo già visto che se la dinamica del segnale non è adattata a quella del quantizzatore, l'SNR è sub-ottimo.  
Perché?
- Risposta formale "analitica"
  - ✓ Il PF si raddoppia perché dimezzando l'ampiezza si dimezza anche  $\sigma_x$
  - ✓ Quindi  $-20 \cdot \log_{10}(2)$  equivale a  $-6$  dB

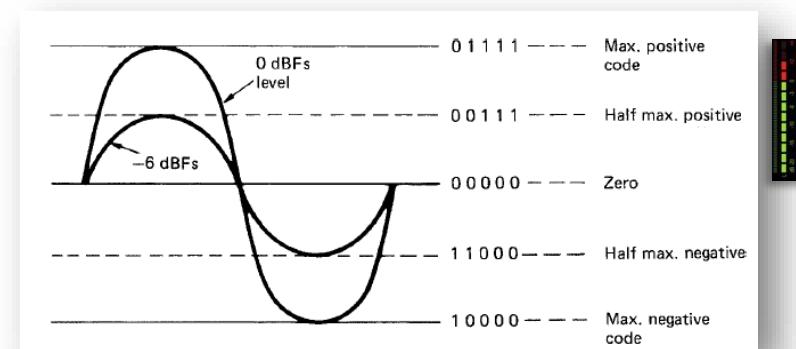
Esempio:  
sinusoide di ampiezza 1 o 0.5



# Dinamica del segnale vs. SNR

## ■ Esempio: quantizzazione di una sinusoide

- ✓ Quantizzatore N bit ed  $x_m=1$
- ✓ Sinusoide di ampiezza A
  - $\sigma = A/\sqrt{2}$



Esempio:  
sinusoide di ampiezza 1 o 0.5

## ■ Calcolo SNR

- ✓  $SNR = +6.02 N - 20 \log_{10} \left( \frac{1}{A/\sqrt{2}} \right) + 4,77 = +6.02N - 3 + 4,77 + 20 \log_{10}(A)$
- ✓  $20 \log_{10}(A) \Rightarrow 0$  per  $A=1$ ,  $-6$  per  $A=1/2$ ,  $-12$  per  $A=1/4$ , ecc.

$$SNR = +6.02 N - 20 \log_{10} \left( \frac{x_m}{\sigma_x} \right) + 4,77$$

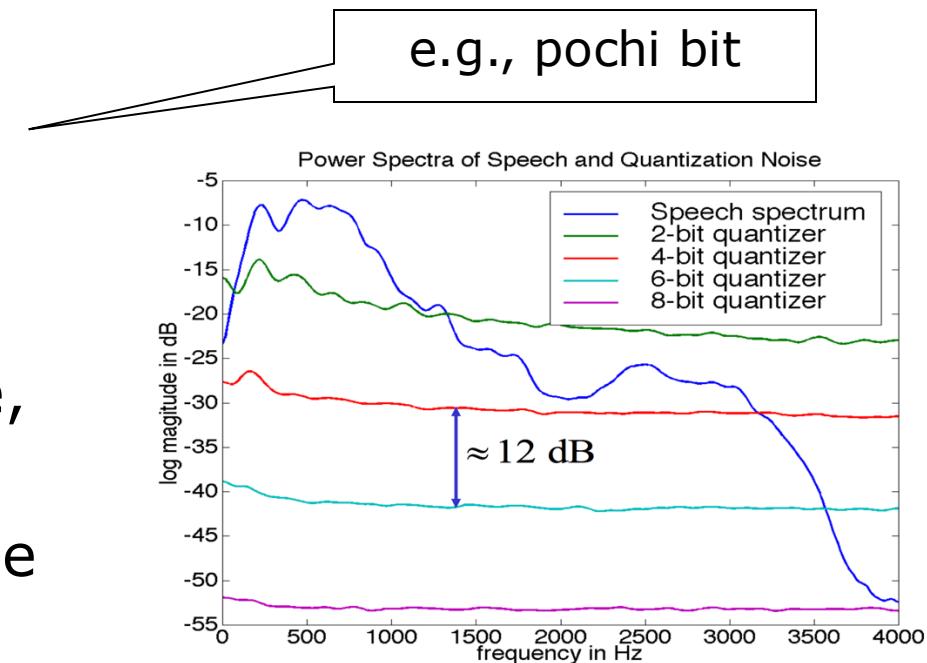
---

**DISTORSIONE ARMONICA**  
errore di quantizzazione correlato con il segnale

# Distorsione (an-)armonica

ASCOLTO

- Quando la quantizzazione è “sufficientemente buona” l’errore di quantizzazione è scorrelato dal segnale
  - ✓ Scorrelato dal segnale = casuale = spettro piatto = rumore bianco
- Altrimenti (quintizzazione NON buona)
  - ✓ L’errore “tende a seguire” lo spettro del segnale
  - ✓ Compaiono delle distorsioni armoniche, anche dette “birdsinging”
  - ✓ Fino a diventare molto simile al segnale



Note: Widrow "under assumptions of high resolution and smooth densities, the quantization error behaves much like random “noise”: it had small correlation with the signal and had approximately a flat (“white”) spectrum."

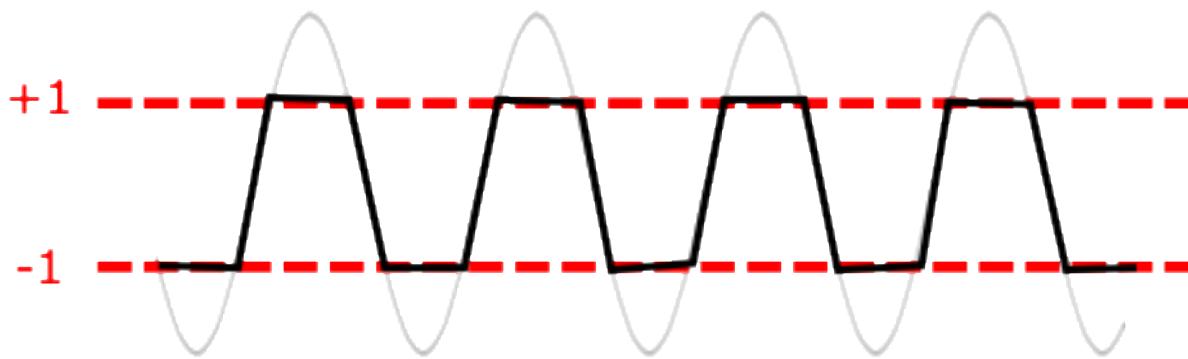
# Distorsione (an-)armonica

- Quando il numero di bit è molto basso   $SNR = +6.02 N$
- Quando il numero di bit è sufficiente ma ...
  - ✓ Il segnale non è adattato al range del quantizzatore   $-20 \log_{10} \left( \frac{X_m}{\sigma_x} \right) +4,77$
- Clipping
  - ✓ Campioni con valori oltre il range del quantizzatore
- Low-level distortion (più subdola)
  - ✓ Ampiezza del segnale compatibile con il passo del Q.  $\Delta Q$
  - ✓ Segnale molto regolare (e.g. code dei suoni)
    - Invece che casuale l'errore diventa periodico e produce armoniche spurie, si percepiscono toni artificiali

# Clipping

---

- Quando il PF è troppo elevato il segnale va oltre il range del quantizzatore per un numero significativo di campioni
- Può accadere
  - ✓ Durante l'acquisizione
  - ✓ Durante il salvataggio dopo l'elaborazione
- Si parla di overload / clipping (tosatura del segnale)
- Domanda: cosa accade allo spettro della (ex) sinusoide?



[AUDACITY DEMO]

# Clipping

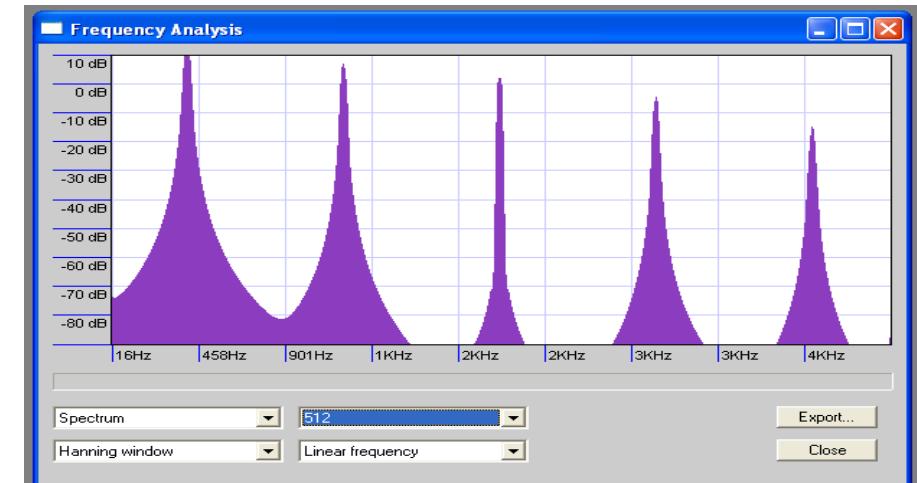
---

- Dimostrazione con Audacity
  - ✓ Si crei un tono puro di ampiezza  $A = 0.8$
  - ✓ Lo si amplifichi di +12 dB (ammettendo il clipping)
  - ✓ Si controlli lo spettro
    - Lo spettro non cambia perché internamente alla DAW i calcoli sono in floating point (non avviene la quantizzazione)
  - ✓ Si forzi la quantizzazione esportando il segnale su file, e.g. .wav
  - ✓ Si carichi il file salvato
  - ✓ Si ascoltino i due segnali e si confrontino gli spettri
- Controllare sempre il clipping prima del salvataggio ed eventualmente normalizzare la dinamica
  - ✓ Audacity > Analyze > Find Clipping

Reference:

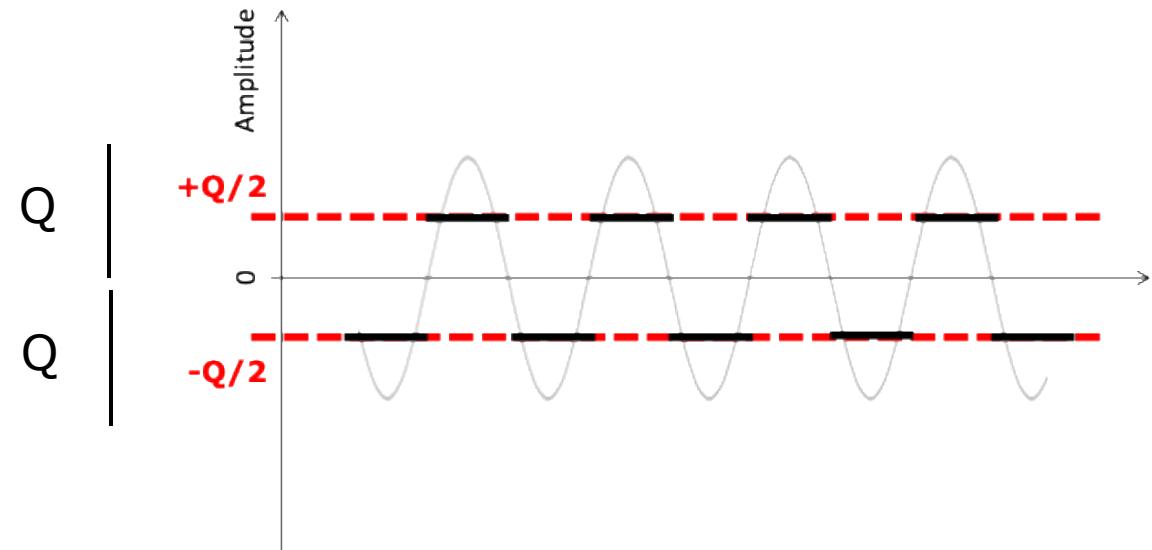
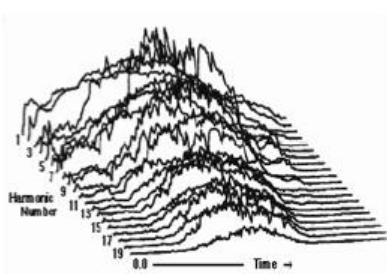
# Clipping => distorsione armonica

- Effetto della distorsione in caso di RMS troppo elevato e clipping
- Si creano delle armoniche che non esistevano nel segnale originale
  - ✓ Errore correlato con il segnale invece che rumore bianco
  - ✓ Ogni componente in frequenza da tono diventa onda quadra
- Da tono ad onda quadra
  - ✓ Si aggiungono le armoniche dispari:  
toni a frequenza multipla intera dispari  
(3, 5, 7 ...) della frequenza originale  
(a intensità decrescente)
  - ✓ Il suono è più aspro  
(causa delle armoniche *dispari*)



# Low-level distortion

- Consideriamo un tono puro (sinusoide) che copre solo due livelli del segnale, quindi "usa" solo un bit del quantizzatore
  - ✓ Assume quindi solo due valori  
 $+Q/2$  se  $> 0$ ,  $-Q/2$  se  $< 0$
  - ✓ Anche in questo caso la sinusoide diventa praticamente una onda quadra
- Esempio: "code" dei suoni



# Distorsione armonica: soluzioni

---

- Quali soluzioni sono possibili per risolvere il problema della distorsione armonica?
- Clipping: ...
- Low-level distortion: ...

# Distorsione armonica: soluzioni

---

- Quali soluzioni sono possibili per risolvere il problema della distorsione armonica?
- Clipping:
  - ✓ La soluzione è semplice basta ridurre o evitare il clipping abbassando il volume in acquisizione o riducendo l'ampiezza prima di salvare
- Low-level distortion:
  - ✓ La soluzione è più complicate e non senza svantaggi
  - ✓ In pratica si vuole trovare un modo per aumentare la precisione del quantizzatore oltre l'LSB (least significant bit)

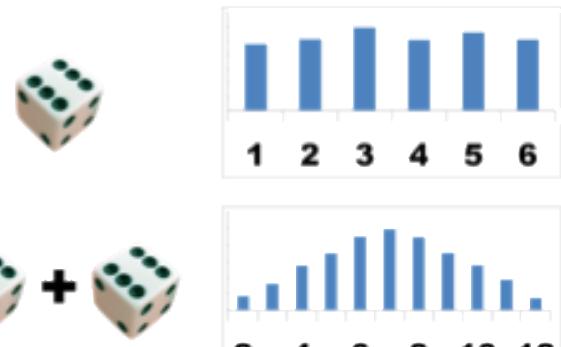
# Dithering

---

- Consiste nell'aggiungere un piccolo rumore casuale al segnale prima della quantizzazione per "spezzare" la regolarità del segnale
  - ✓ Trasforma la distorsione periodica (armoniche spurie) in rumore stocastico (bianco) meno fastidioso all'orecchio
  - ✓ Il prezzo da pagare è una lieve riduzione dell'SNR



# Dithering - dettagli



- Caratteristiche rumore casuale
  - ✓ Più efficace rumore "triangolare" rispetto a quello "piatto" (uniforme)
    - Si ottiene sommando due distribuzioni uniformi (e.g. la somma di due dadi)
  - ✓ Valori tra  $[-\Delta Q, +\Delta Q]$
  - ✓ Peggioramento dell'SNR dell'ordine di 3 dB, ma suona più piacevole



N.B. Tecniche più raffinate introducono "noise shaping" che concentra il rumore nelle frequenze meno percepibili.

# Dithering - demo

---

- Ascolto ./media/dither

- ✓ reaper-sine-wave-dithering.part-end-tpdf.ead25.mp4
- ✓ quantization-3bit-with-without-dither.wav
- ✓ quantization-2bit-with-without-dither.wav

---

MEETING

# Calcolo del "volume" o livello

---

- Ogni dispositivo / programma audio che voglia dirsi tale è bene che visualizzi il "livello" dell'audio acquisito o riprodotto
- Considerando un "frame" (intervallo di N campioni consecutivi) possiamo misurare
  - ✓ *il valore di picco*: massimo dei valori assoluti dei campioni
  - ✓ *il valore medio*: media dei valori dei campioni (tendenzialmente zero)
  - ✓ *il valore efficace*: radice quadrata della media dei valori al quadrato
    - Rispetto alla media dei valori assoluti è più legata al concetto fisico di energia e alla percezione (pesa di più i picchi)

*Volume* o *loudness* è da considerarsi, propriamente, riferito al "livello" sonoro percepito dal nostro orecchio (più che quello misurato) e quindi pesato percettivamente.

---

# Calcolo del valore efficace

---

- Per valore efficace si intende l'RMS (Root Mean Square) value
  - ✓ E' l'equivalente digitale della pressione efficace

$$RMS = \sqrt{\frac{1}{n} \sum_{i=0}^{n-1} x_i^2}$$

$$p_{\text{eff}} = \left[ \frac{1}{T} \int_0^T [\Delta p(t)]^2 dt \right]^{\frac{1}{2}}$$

- ✓ Espresso in dB è l'equivalente digitale del livello di pressione sonora

$$L_p = 10 \log \left( \frac{p}{p_{\text{ref}}} \right)^2$$

- Cosa serve per calcolarlo?

# Calcolo del valore efficace

---

- Per valore efficace si intende l'RMS (Root Mean Square) value
  - ✓ E' l'equivalente digitale della pressione efficace

$$RMS = \sqrt{\frac{1}{n} \sum_{i=0}^{n-1} x_i^2}$$
$$p_{\text{eff}} = \left[ \frac{1}{T} \int_0^T [\Delta p(t)]^2 dt \right]^{\frac{1}{2}}$$

- ✓ Espresso in dB è l'equivalente digitale del livello di pressione sonora

$$L_p = 10 \log \left( \frac{p}{p_{\text{ref}}} \right)^2$$

- Cosa serve per calcolarlo?
  - ✓ Serve un intervallo di integrazione. Piccolo o grande?
  - ✓ Serve un valore di riferimento. Quale?

# Calcolo del valore efficace

---

## ■ RMS (Root Mean Square) value

✓ Dato il segnale  $x$

```
def RMS(x):  
    return np.sqrt(np.mean(x**2))  
  
w, t = generate_sinusoid(amp=1)  
RMS(w)
```

$$0.707 \quad \text{amp}/\sqrt{2}$$

$$RMS = \sqrt{\frac{1}{n} \sum_{i=0}^{n-1} x_i^2}$$

$$P_{\text{eff}} = \left[ \frac{1}{T} \int_0^T [\Delta p(t)]^2 dt \right]^{\frac{1}{2}}$$

■ Per trasformarlo l'RMS in dB abbiamo bisogno di un riferimento (come la  $p_{\text{ref}}$  per  $p$ ): usiamo una  $A_{\text{ref}}$  o meglio un  $RMS_{\text{ref}}$

$$L_p = 10 \log \left( \frac{p}{p_{\text{ref}}} \right)^2$$

# dB<sub>FS</sub>: Full Scale

$$L_p = 10 \log\left(\frac{p}{p_{ref}}\right)^2 dB_{SPL}$$

- Nell'audio digitale il livello misurato in dB è sempre riferito al livello massimo del segnale
  - ✓ Per convenzione, se l'audio (A) è compreso tra -1 e +1 il valore |A<sub>ref</sub>| massimo è 1
  - ✓ Di conseguenza il livello RMS<sub>FS</sub> massimo è 1
  - ✓ Quindi **RMS dB<sub>FS</sub>= 20 log<sub>10</sub> (RMS/RMS<sub>FS</sub>)**
  - ✓ **0 dB<sub>FS</sub>** corrisponde quindi al segnale **con valore massimo (full scale)**

**N.B. a differenza di dB<sub>SPL</sub> il livello dB<sub>FS</sub> è sempre NEGATIVO**

*perchè RMS è sempre minore o uguale di RMS<sub>FS</sub>*

# dB<sub>FS</sub>: Full Scale

$$L_p = 10 \log\left(\frac{p}{p_{ref}}\right)^2 dB_{SPL}$$

- Nell'audio digitale il livello misurato in dB è sempre riferito al livello massimo del segnale

- ✓ Per convenzione, se l'audio (A) è compreso tra -1 e +1 il valore  $|A_{ref}|$  massimo è 1
  - ✓ Di conseguenza il livello RMS<sub>FS</sub> massimo è 1
  - ✓ Quindi **RMS dB<sub>FS</sub>= 20 log<sub>10</sub> (RMS/RMS<sub>FS</sub>)**
  - ✓ **0 dB<sub>FS</sub>** corrisponde quindi al segnale **con valore massimo (full scale)**

- ✓ Esempio: per una sinusoide di ampiezza 1 RMS dBFS=  $20 \log_{10} (0.707/1) = -3$  dB

- ✓ In generale per una sinusoide

$$RMS_{dB_{FS}}\{\sin(A)\} = 20 \log_{10} \left( \frac{A/\sqrt{2}}{1} \right)$$

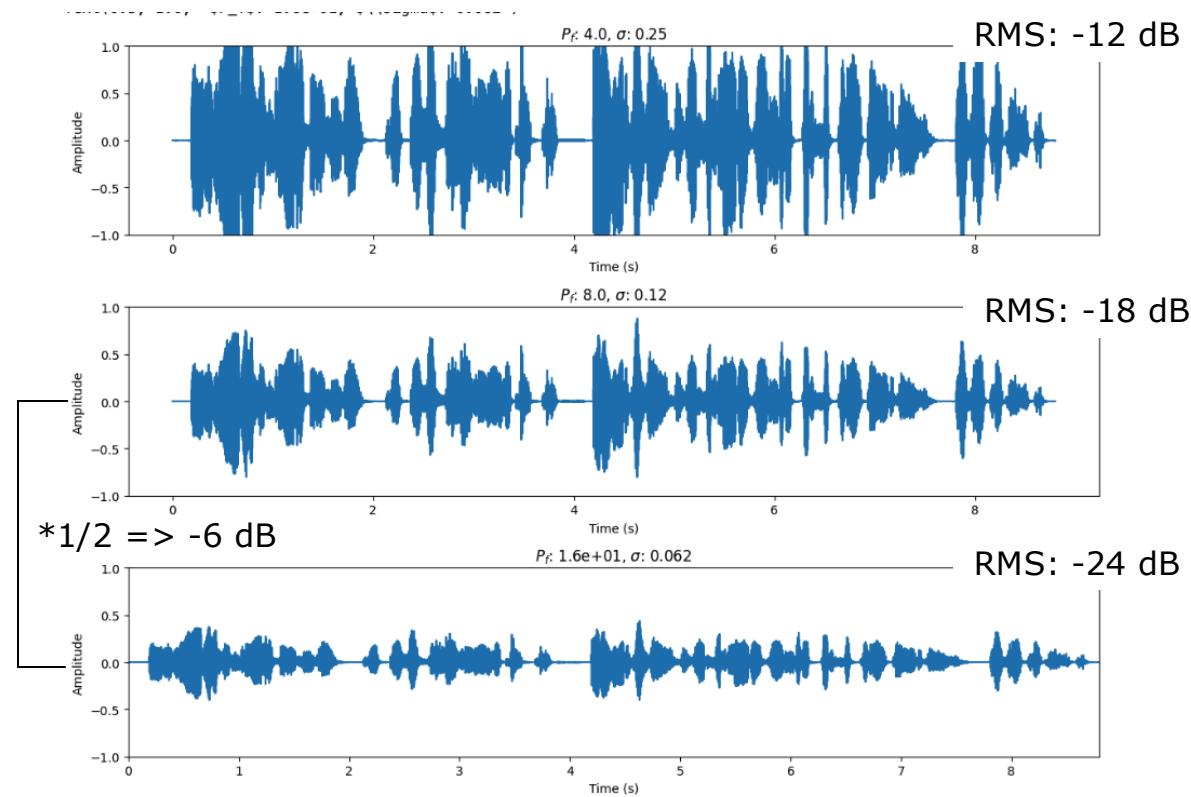
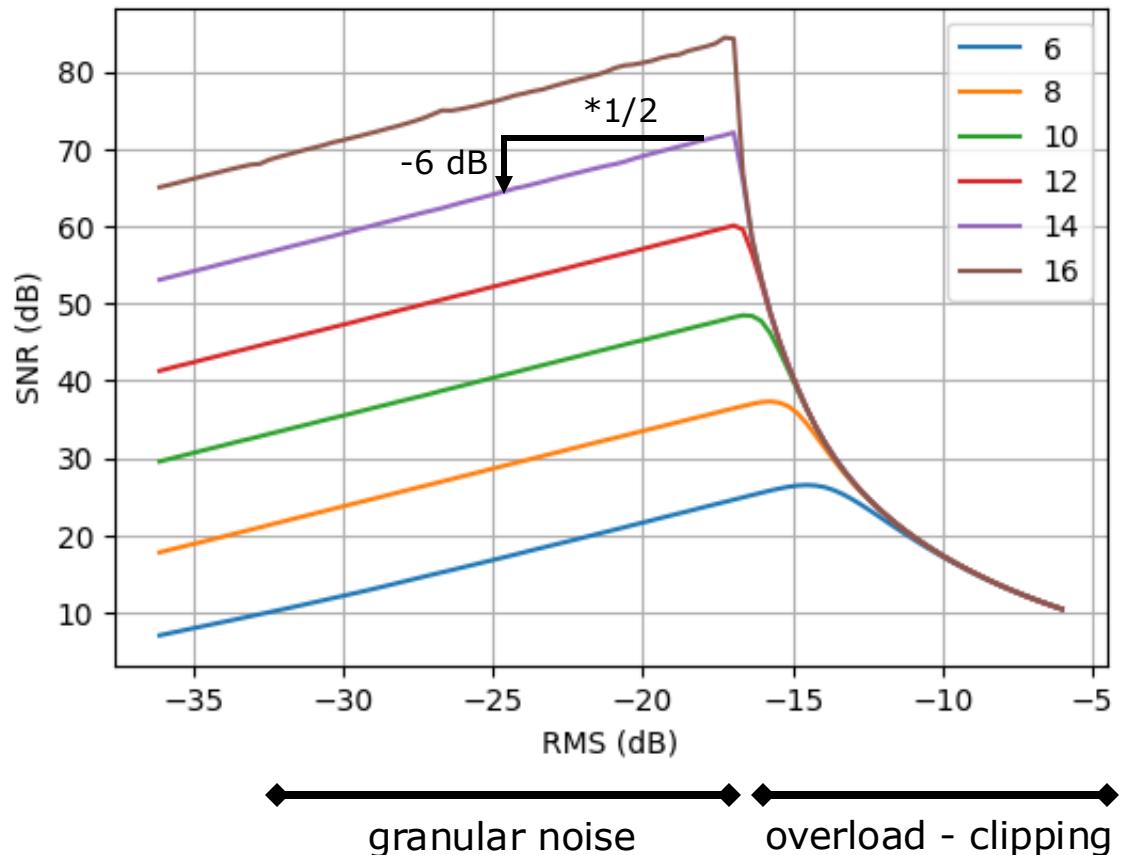
**N.B. a differenza di dB<sub>SPL</sub> il livello dB<sub>FS</sub> è sempre NEGATIVO**

*perchè RMS è sempre minore o uguale di RMS<sub>FS</sub>*

# Esercizio: soluzione

$$RMS_{dB_{FS}}(x[n]) = -20 \log_{10}(\sigma_x / 1)$$

- Come varia l'SNR in funzione di  $\sigma$  per un brano vocale  $x$ , fissato N (numero di bit)

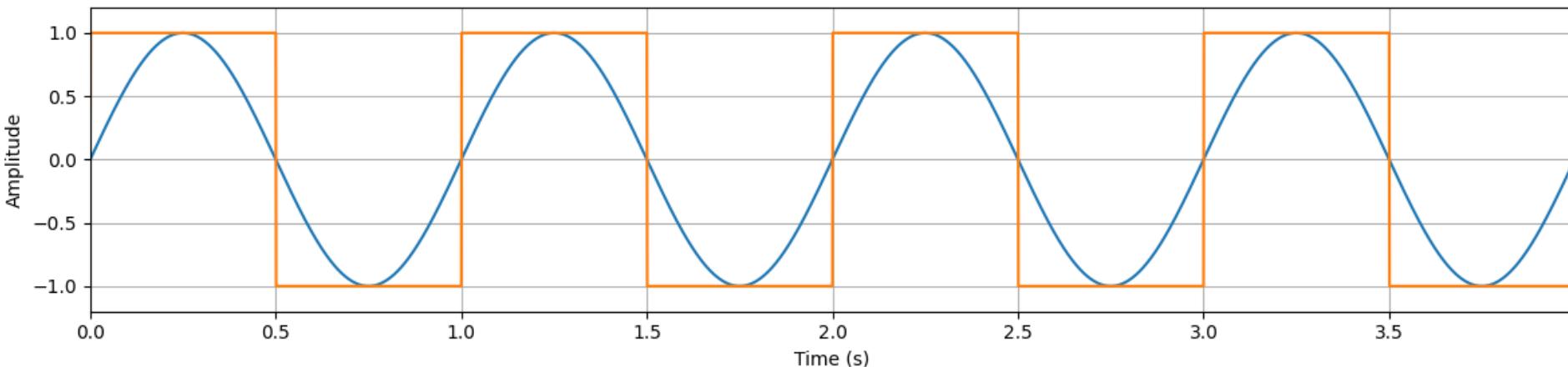


Gsheet: ead26.quantizer\_snr\_vs\_rms

# Esercizio dB<sub>FS</sub>

---

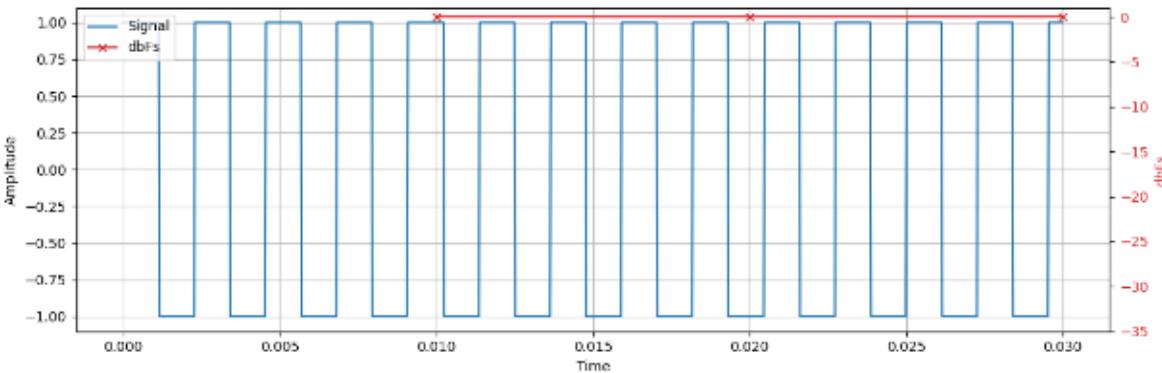
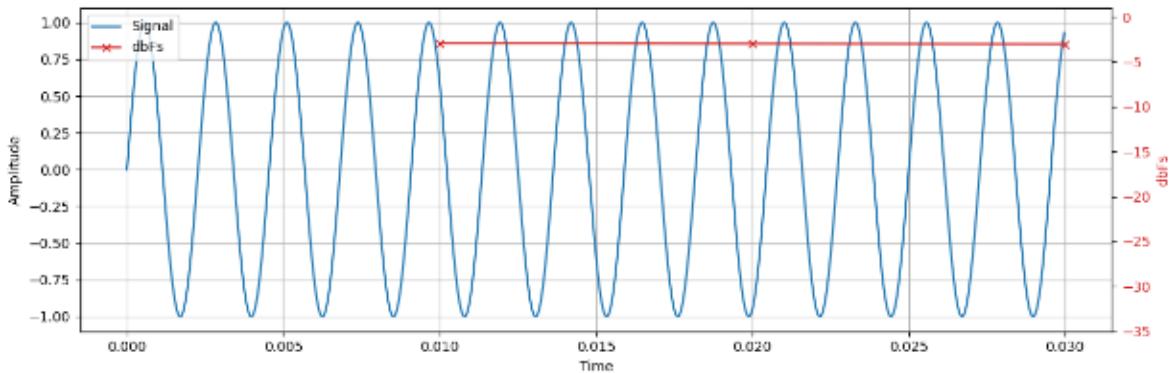
- Un'onda quadra può essere derivata "raddrizzando" la sinusoida cioè derivandone semplicemente il segno  
+1 quando positiva, -1 quando negativa
  - ✓ Generate un'onda quadra di ampiezza 1 e 0.5 e calcolatene il livello dB<sub>FS</sub>
  - ✓ E' maggiore o minore di quello della corrispondente sinusoida?



# Esercizio dB<sub>FS</sub>

- Un'onda quadra può essere derivata "raddrizzando" la sinusoide cioè derivandone semplicemente il segno +1 quando positiva, -1 quando negativa
  - ✓ Generate un'onda quadra di ampiezza 1 e 0.5 e calcolatene il livello dB<sub>FS</sub>
  - ✓ E' maggiore o minore di quello della corrispondente sinusoide?

N.B. L'AES e l'ITU associano invece il livello di 0 dB FS alla sinusoide di ampiezza 1, l'onda quadra avrebbe quindi +3dB FS (ma in questo corso non facciamo così)

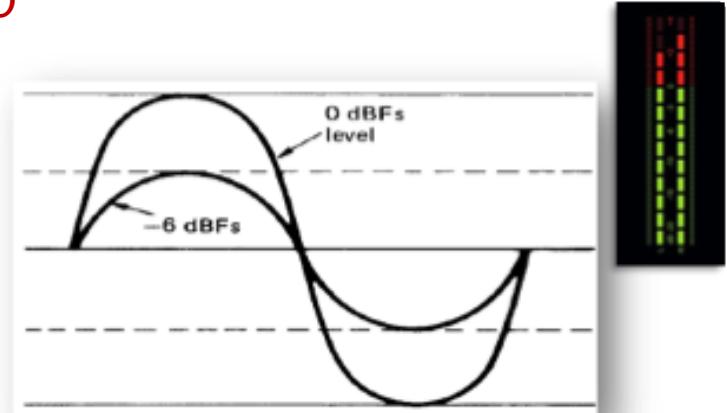


# dB<sub>FS</sub>: Full Scale

- Relazione db<sub>FS</sub> ed ampiezza di un segnale
- Ricordando che  $RMS(A \cdot x) = |A| \cdot RMS(x)$ 
  - ✓ Se moltiplichiamo per A un segnale x, l'RMS scala in proporzione, quindi anche il livello dbFS

$$RMS_{dB_{FS}} = 20 \log_{10} \left( \frac{A \cdot RMS_x}{RMS_{FS}} \right) = 20 \underbrace{\log_{10}(A)}_{\text{ricordarsi che è un logaritmo}} + 20 \log_{10} \left( \frac{RMS_x}{RMS_{FS}} \right)$$

- ✓ Quindi se  $A=1/2$  il livello dB<sub>FS</sub> del segnale diminuisce di  $20 \log_{10}(1/2) = -6 \text{dB}_{FS}$



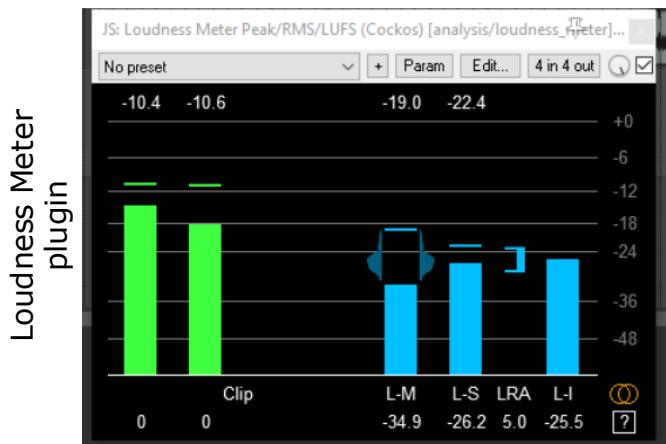
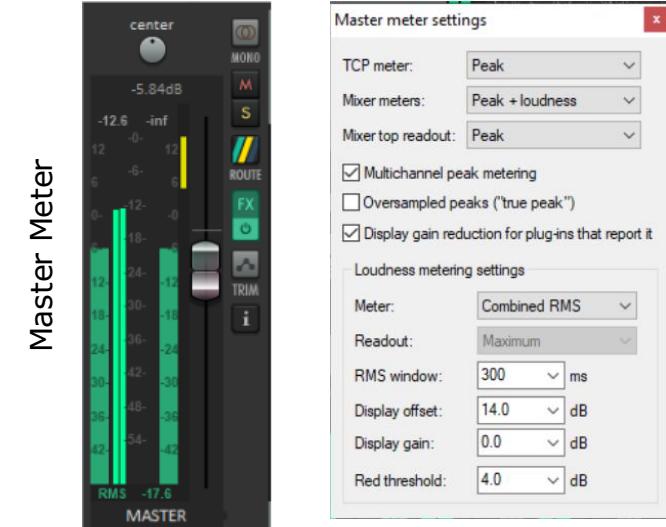
# Reaper VU Meter (volume unit)

## ■ Impostazioni (right click)

- ✓ TCP (track control panel) meter
- ✓ Mixer meter\_s: centrali e laterali
  - Peak (PPM, Peak Program Meter)  
[https://en.wikipedia.org/wiki/Peak\\_programme\\_meter](https://en.wikipedia.org/wiki/Peak_programme_meter)
  - Loudness (RMS, LUFS) <https://en.wikipedia.org/wiki/LUFS>
- ✓ Tempo di integrazione (RMS window)
  - Intervallo su cui è calcolato l'RMS (o il picco)

## ■ Approfondimento:

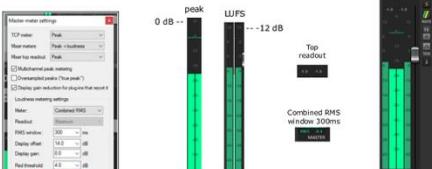
AES – Loudness <https://aes2.org/audio-topics/loudness-2/>  
<https://aes2.org/resources/audio-topics/loudness-project/learn-more/>



# LUFS

## Metering - Reaper

- Master meter settings (RUG 11.10)
  - ✓ tasto destro sul meter



DAW - Editing Audio

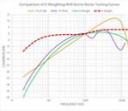
## LUFS (aka LKFS)

- Standard ITU-R BS.1770 / EBU R128, per broadcast, streaming, ...
  - ✓ e.g. streaming -14 LUFS, broadcast -23 LUFS, download: -10 LUFS
- LUFS (*Loudness Units relative to Full Scale*)
  - ✓ Molto simili ai dBFS ma con ponderazione psicoacustica (K-weighting) per tenere conto della sensibilità dell'orecchio umano.
- Tipi di LUFS
  - ✓ Momentary LUFS-M → finestra breve ( $\approx 400$  ms), variazioni rapide
  - ✓ Short-term LUFS-S → finestra media ( $\approx 3$  s), andamento locale
  - ✓ Integrated LUFS-I → intero brano, standard di normalizzazione
    - Usa GATING per scartare rumore di fondo e parti poco udibili

DAW - Editing Audio

## AES Loudness Resource

- Basics, Normalization, Insights from professionals
  - ✓ <https://aes2.org/resources/audio-topics/loudness-2>
- Learn more
  - ✓ <https://aes2.org/resources/audio-topics/loudness-project/learn-more/>
  - ✓ Peak Metering, True Peaks, Clipping Distortion
  - ✓ How is RMS computed and how to interpret it?
  - ✓ K-weighting (broadcast): HP + head effect
  - ✓ How the Loudness Meter Works



DAW - Editing Audio

## Range dinamico (audio)

- Intuitivamente è la differenza tra i livelli più bassi e i livelli più alti del segnale audio, espressa in dB
  - ✓ N.B. differente dal range dinamico *digitale*, e.g. del quantizzatore che è "solo" il rapporto in dB tra il valore massimo e il valore minimo
- Audio Dynamic Range – EBU/ITU
  - ✓ LRA (Loudness Range) è calcolato in LUFS =  $L_{10} - L_{95}$ 
    - Differenza tra il 10° e 95° percentile dei livelli di short-time loudness
    - I livelli di LUFS<sub>S</sub> sono calcolati su finestre di 3s sovrapposte per 2s
  - ✓ Il filtraggio / gating  $G_a = G_r - 20$  LU (diverso da calcolo LUFS<sub>I</sub>)

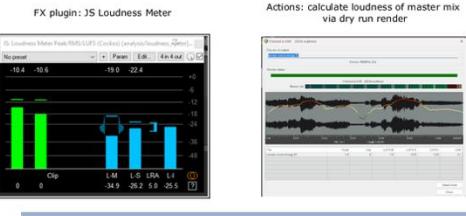
Reference: [https://www.itu.int/dms\\_pubrec/itu-r/rec/bs/R-REC-BS.1770-3-201208-S!!PDF-E.pdf](https://www.itu.int/dms_pubrec/itu-r/rec/bs/R-REC-BS.1770-3-201208-S!!PDF-E.pdf)

Approfondimento => "Calcolo LUFS"

DAW - Editing Audio

30

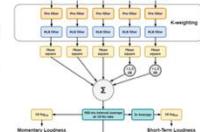
## Analisi LUFS



DAW - Editing Audio

## Calcolo Integrated LUFS

- Pre-processing: vettore di K-weighted RMS su 400ms/10Hz (RMS<sub>K</sub>)
  - ✓ K-weighting filter
  - ✓ Calcolo mean square per ogni frame e canale
  - ✓ Somma pesata dei valori
- Da RMS<sub>K</sub> si può calcolare
  - ✓ Momentary Loudness =  $10 \cdot \log_{10}(RMS_K)$
  - ✓ Short-Term Loudness = idem, ma mediante RMS<sub>K</sub> su 3 secondi

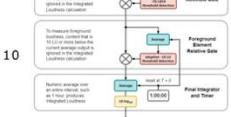


Reference: <https://aes2.org/resources/audio-topics/loudness-project/learn-more/>

DAW - Editing Audio

## Calcolo Integrated LUFS

- 2 fasi x stimare e scartare il rumore di fondo e le parti poco udibili
- Fase assoluta livello  $G_a = -70$  LUFS
  - ✓ I valori RMS<sub>K</sub> al di sotto di  $G_a$  vengono scartati
- Fase relativa livello  $G_r$ 
  - ✓ Dato  $L_a$  il valore in dB della media dei valori rimanenti di RMS<sub>K</sub>
  - ✓ I valori RMS<sub>K</sub> al di sotto di  $G_r = L_a - 10$  vengono scartati
- LUFS è il valore in dB della media dei valori rimanenti



Reference: <https://aes2.org/resources/audio-topics/loudness-project/learn-more/>

DAW - Editing Audio

## Range dinamico (audio) - esempio

```
% Apply the absolute-threshold gating (non-recessive definition)
abs_gate_vec = (ShortTermLoudness >= ABS_THRESHOLD);
% abs_gate_vec is a vector of loudness levels above absolute threshold
% abs_gate_vec = 1 if short term loudness is above absolute threshold
% else abs_gate_vec = 0. The result is a binary vector
% only include loudness levels that are above gate threshold
st_gate_vec = st_abs_gate_and_vec(abs_gate_vec);

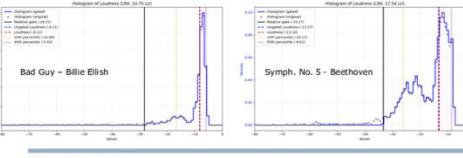
% Apply the relative-threshold gating (non-recessive definition)
n = length(st_abs_gate_and_vec);
st_gate_vec = st_abs_gate_and_vec * (st_abs_gate_and_vec >= REL_THRESHOLD);
% st_gate_vec is a vector of loudness levels above relative threshold
% st_gate_vec = 1 if short term loudness is above relative threshold
% else st_gate_vec = 0. The result is a binary vector
% only include loudness levels that are above gate threshold
rel_gate_vec = st_gate_vec >= st_abs_gate_and_vec;
```



Reference: EBU – TECH 3342

## Range dinamico (audio) - esercizio

- Calcolare LUFS e DR con PyLoudNorm modificato
  - ✓ PyLoudNorm J Reiss Queen Mary Univ
  - ✓ Video <https://www.youtube.com/watch?v=krSjPQ3d4gE>
- Visualizzare la distribuzione e i percentili per "stili" diversi



DAW - Editing Audio

49

---

## SINTESI AUDIO - numerica

# Sintesi audio

---

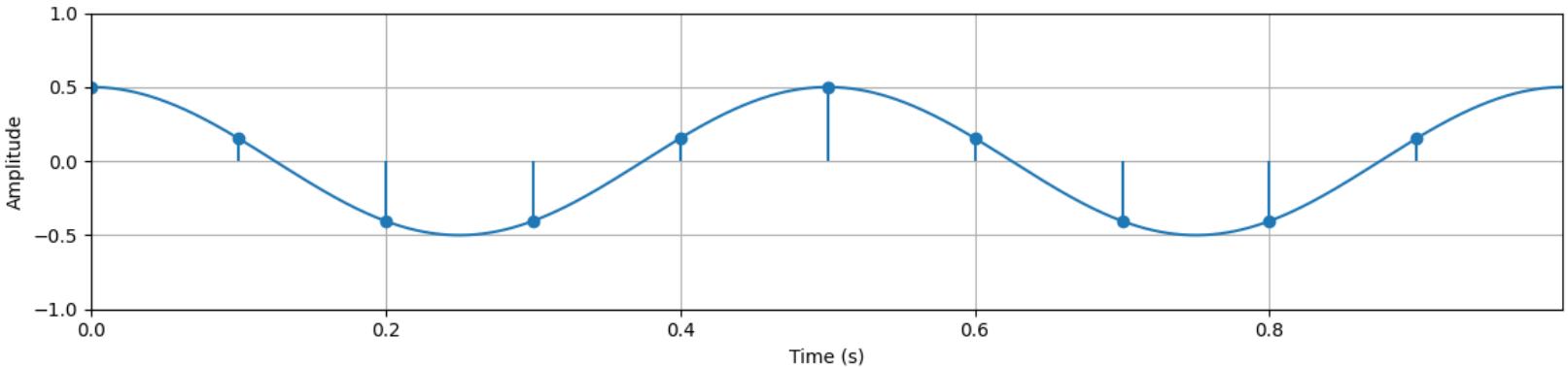
- Qualsiasi sequenza di valori (tra -1 e +1) riprodotta ad una certa velocità genera un segnale audio (eventualmente udibile)
- Una sequenza di valori costanti "non suona"
  - ✓ `IPython.display.Audio( [0.5] * Fs, rate=Fs )`
- Una sequenza di numeri casuali suona come rumore
  - ✓ `IPython.display.Audio( np.random.uniform(-1, 1, int(Fs*d)), rate=Fs )`

Reference:

---

# Sintesi (co-)sinusoide – tono puro

- Una sinusoide a frequenza  $f$  genera un tono puro
  - ✓  $w = A \cdot \cos(2\pi \cdot f \cdot t)$
- Come generare una sinusoide digitale?
  - ✓  $A = 0.5, f = 2$



Esempio didattico: usiamo frequenze molto basse, non udibili, ma più facili da visualizzare e con cui fare i calcoli.

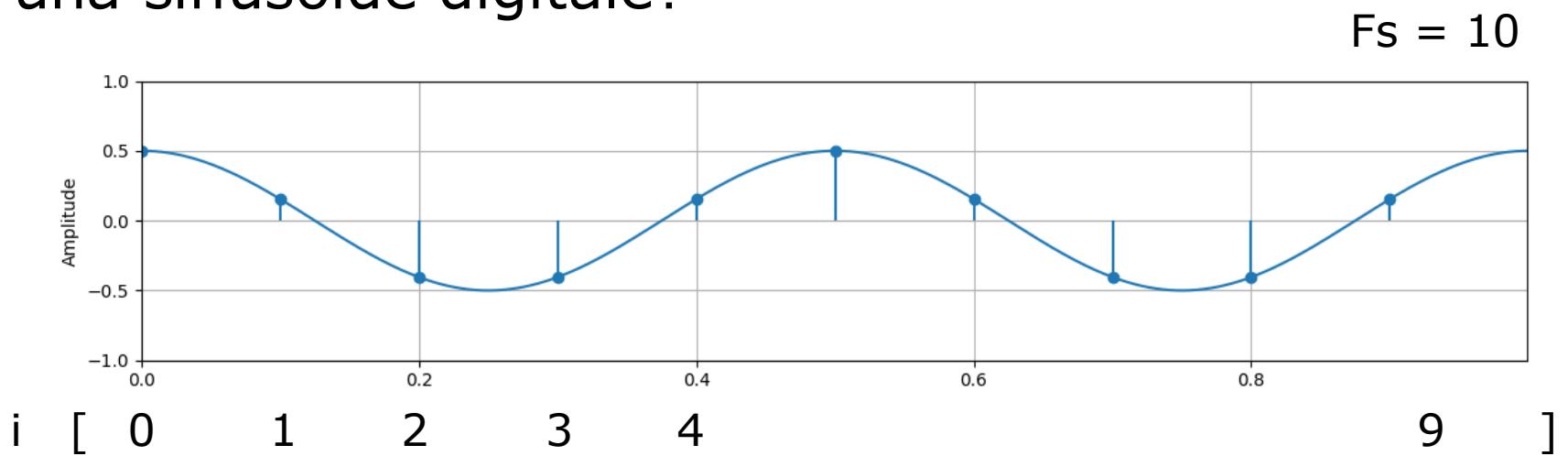
# Sintesi (co-)sinusoide – tono puro

- Una sinusoide a frequenza  $f$  genera un tono puro

✓  $w = A \cdot \cos(2\pi \cdot f \cdot t)$

- Come generare una sinusoide digitale?

✓  $A = 0.5, f = 2$



- Occorre valutare il coseno negli istanti di campionamento

$$\hat{t} = i \cdot 1/F_s$$

Esempio didattico: usiamo frequenze molto basse, non udibili, ma più facili da visualizzare e con cui fare i calcoli.

# Sintesi tono puro

## ■ Generazione degli istanti di campionamento

- ✓ Genero N punti a distanza  $1/F_s$

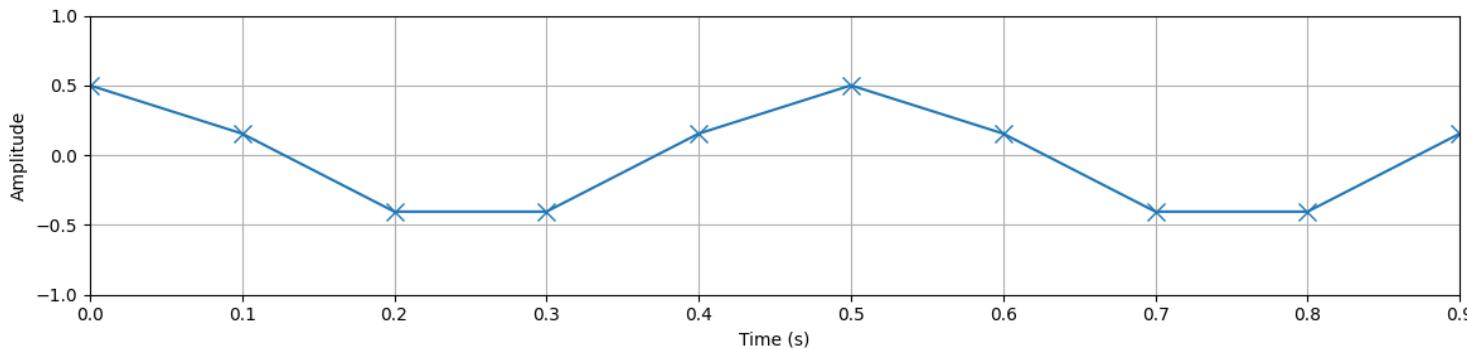
- $t = np.arange(N) * 1/F_s$

Ipotizzando  $N = 10$  e  $F_s = 1/10$  si avrebbe

$$t = [0, 1, 2, 3, 4, 5, 6, 7, 8, 9] * 1/10 \Rightarrow [0, 0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9]$$

- ✓ La durata è data dal numero di campioni  $N$  per la distanza tra gli stessi  $\Delta t$  quindi  $N * 1/F_s$  (secondi)

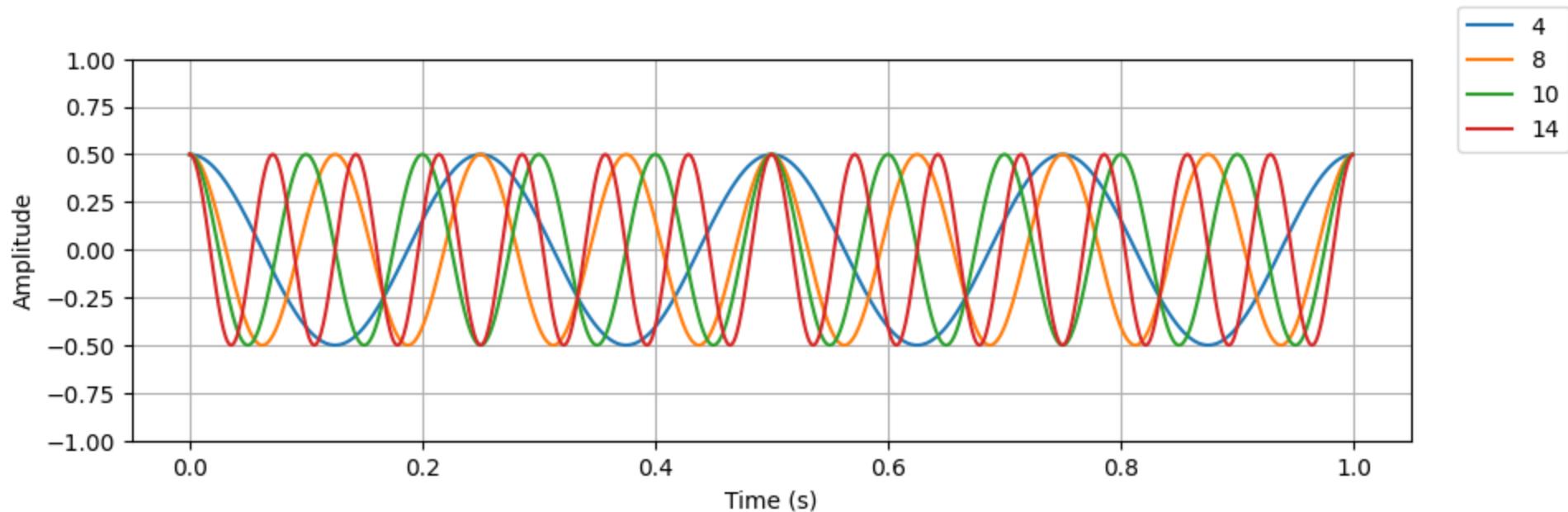
```
import numpy as np
N = 10
Fs = 10
A = 0.5
f = 2
t = np.arange(N) / Fs
s = A * np.cos(2 * np.pi * f * t)
```



# Esercizio

## ■ Domanda

- ✓ Continuiamo a supporre una frequenza di campionamento di 10 Hz
- ✓ Quale forma hanno le sinusoidi corrispondenti alle frequenze di
  - 4, 8, 10, 14 Hz ?

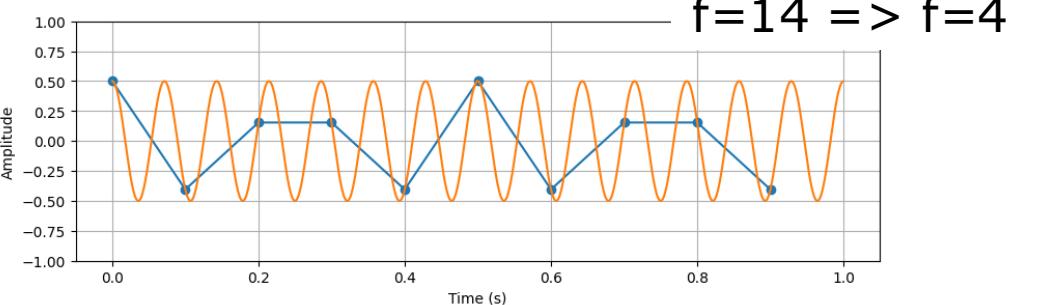
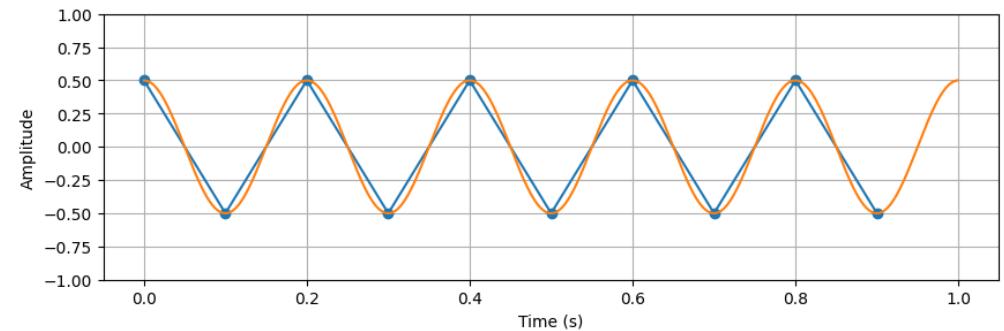
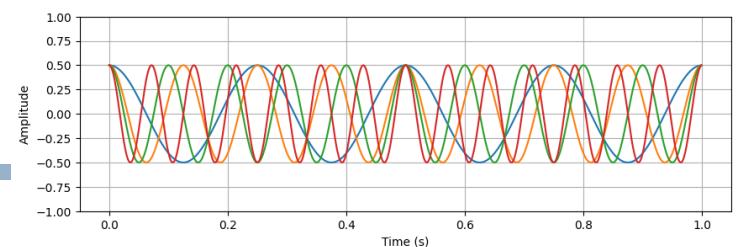
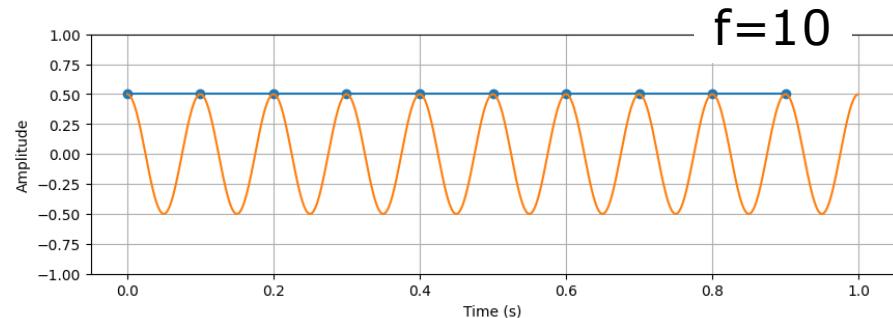


# Esercizio - aliasing

## ■ Domanda

- ✓ Continuiamo a supporre una frequenza di campionamento di 10 Hz
- ✓ Quale forma hanno le sinusoidi corrispondenti alle frequenze di
  - 4, 8, 10, 14 Hz ?

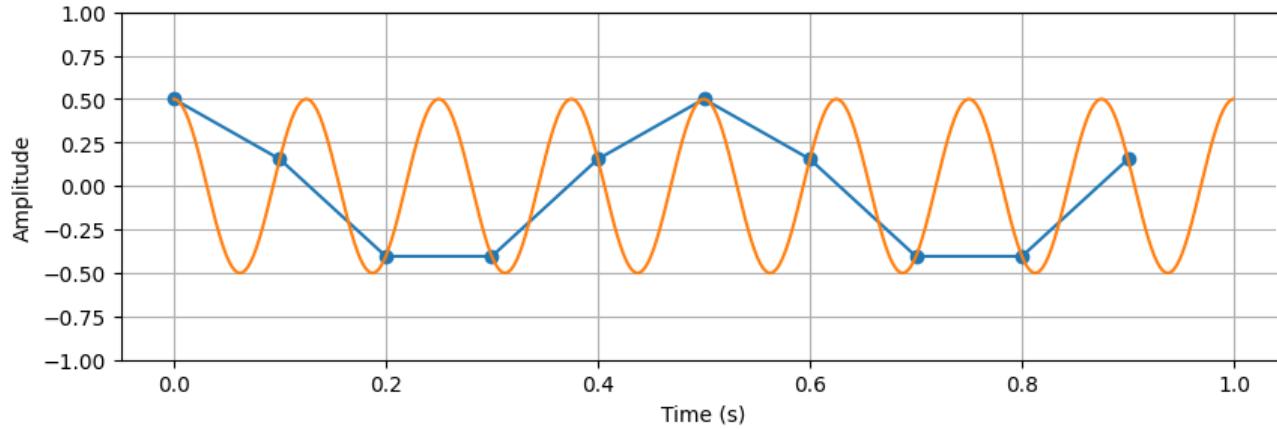
■ Con  $F_s = 10$  Hz non è possibile rappresentare frequenze superiori alla frequenza di Nyquist  $F_s/2 = 5$  che ha 2 campioni per periodo



$f=8$  ?

# Esercizio - foldover

- Le frequenze tra  $F_s$  e  $F_s + F_s/2$
  - Le frequenze tra  $F_s/2$  ed  $F_s$  subiscono foldover
    - ✓ Indicando con  $f$  la frequenza di sintesi e  $f'$  la frequenza risultante
    - ✓  $f/f' = [0/0 \ 1/1 \ 2/2 \ 3/3 \ 4/4 \ 5/5 \ 6/-4 \ 7/-3 \ 8/-2 \ 9/-1 \ 10/0 \ 11/1 \dots]$
- A "specchio" negative**



$f=8 <> f=-2$

```
import numpy as np
N, Fs, A, f = 10, 10, 0.5, 5
t = np.arange(N) / Fs
s = A * np.cos(2 * np.pi * f * t)
```

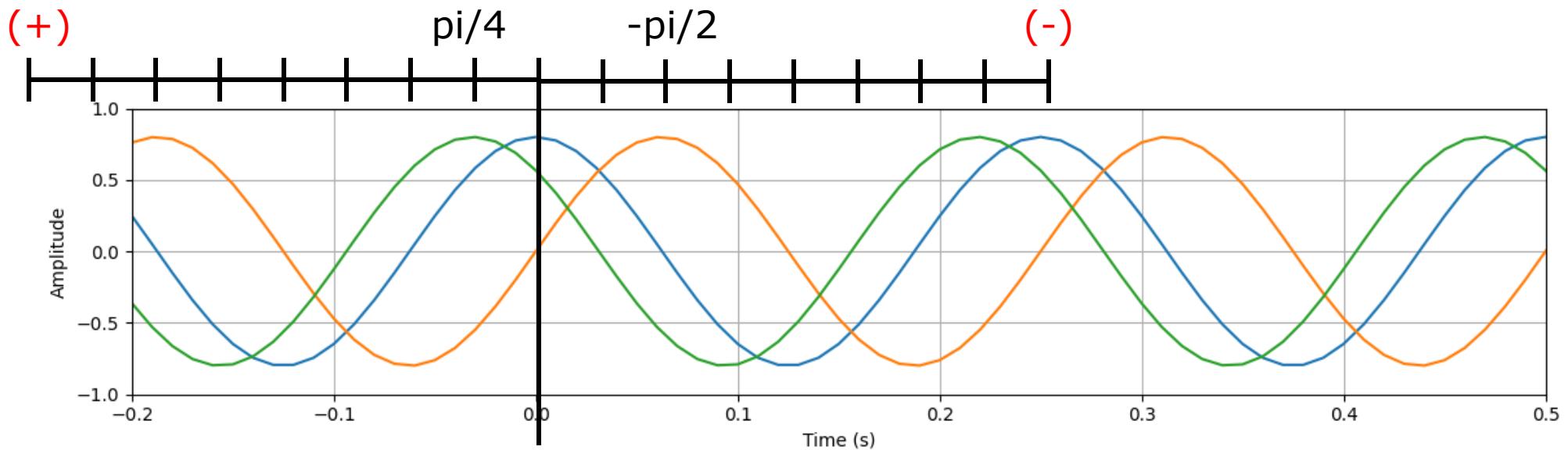
# Effetto della fase

---

- Seno e coseno differiscono per la fase iniziale
  - ✓  $\cos(x) = \sin(x+\pi/2)$  e  $\sin(x) = \cos(x-\pi/2)$
- L'orecchio non è sensibile alla fase assoluta
  - ✓ Il tono (puro) di un seno o di un coseno è indifferente
  - ✓ Diventa sensibile nei segnali complessi se la fase ha un effetto significativo sull'inviluppo temporale del segnale
    - Caso classico somma di microfoni in controfase

# Ritardo di fase

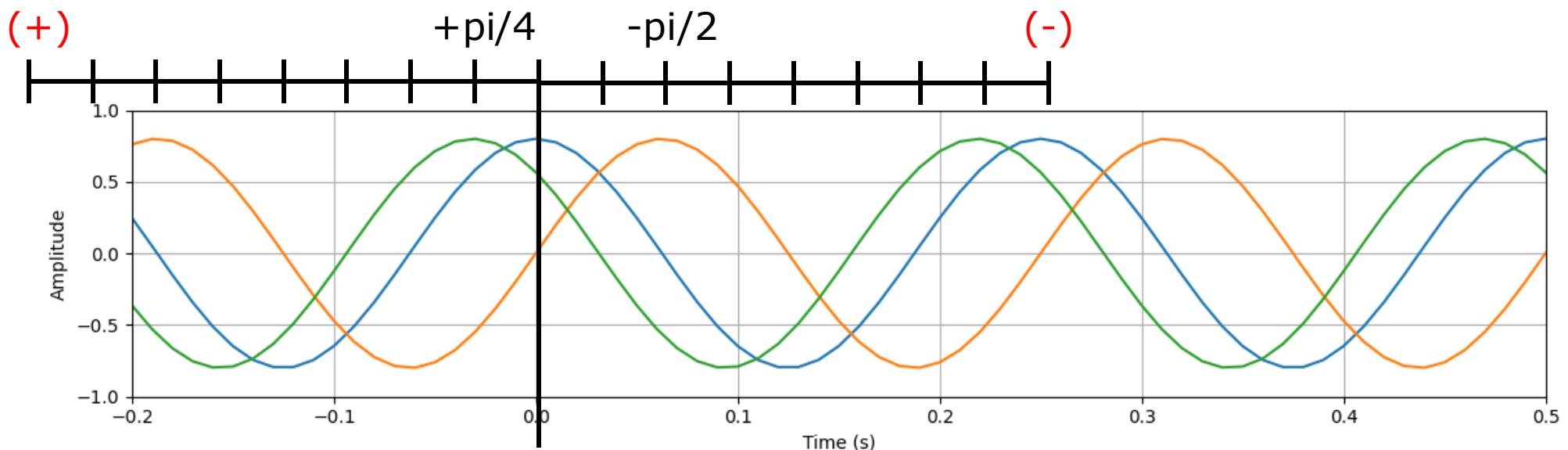
- Si può identificare la fase di un coseno localizzando il picco più vicino allo 0 e invertendone il segno
  - ✓ verde:  $\cos(2\pi f + \pi/4)$
  - ✓ arancio:  $\cos(2\pi f - \pi/2)$



# Ritardo temporale

- Il ritardo di fase può anche essere visto come un ritardo temporale  
 $A \cos(2\pi f(t - t_0))$

- ✓ arancio: picco più vicino dopo lo zero "in ritardo",  $t_0$  positivo
  - ✓ verde: picco più vicino prima dello zero "in anticipo",  $t_0$  negativo
- la relazione tra ritardo di fase e ritardo temporale ha il segno opposto*



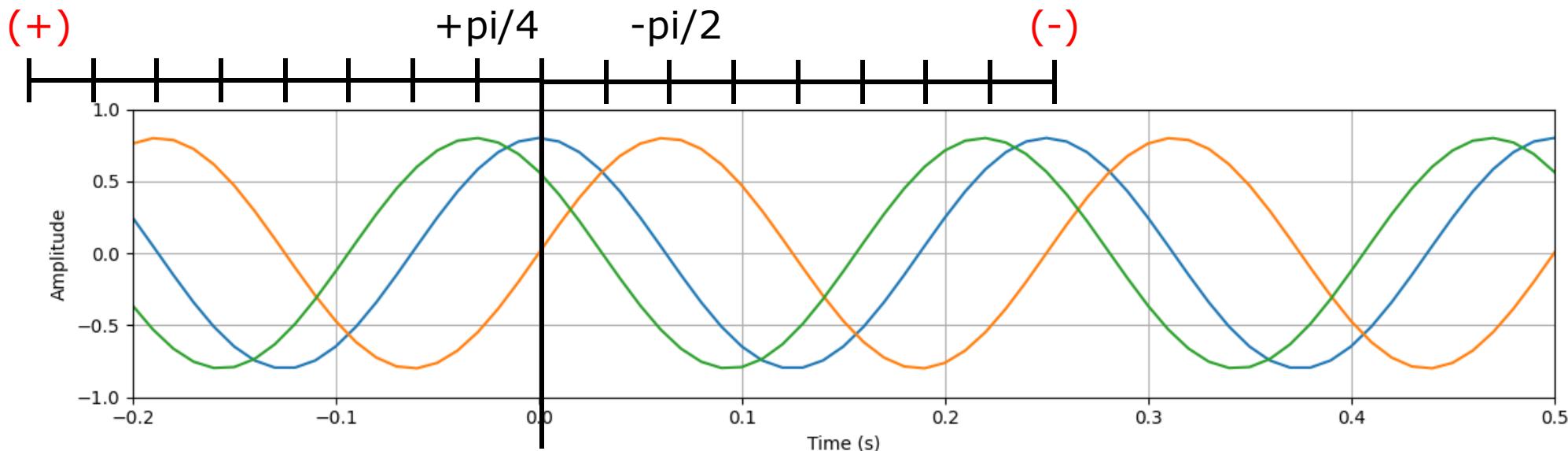
# Ritardo temporale vs ritardo di fase

- Uguagliando le espressioni si deriva la relazione

$$A \cos(2\pi f t + \varphi_0) = A \cos(2\pi f(t - t_0)) \Rightarrow t_0 = -\frac{\varphi}{2\pi f}$$

A parità di ritardo di fase  
il ritardo temporale è  
funzione della frequenza / periodo.

- In funzione del periodo  $T = 1/f \Rightarrow t_0 = -\frac{\varphi}{2\pi} T$



# Ritardo temporale vs ritardo di fase

- Uguagliando le espressioni si deriva la relazione

$$A \cos(2\pi f t + \varphi_0) = A \cos(2\pi f(t - t_0)) \Rightarrow t_0 = -\frac{\varphi}{2\pi f}$$

A parità di ritardo di fase  
il ritardo temporale è  
funzione della frequenza  
/ periodo.

- In funzione del periodo  $T = 1/f \Rightarrow t_0 = -\frac{\varphi}{2\pi} T$

- Esempio

- ✓ Dato un ritardo di fase di  $-\pi/4$  il ritardo temporale dipende
- ✓ In generale è positivo e  $(\pi/4 / 2*\pi) = 1/8$  del periodo
  - $f = 10, T = 0.1 \Rightarrow 0.0125$
  - $f = 5, T = 0.2 \Rightarrow 0.025$

Vedi problema relazione di fase  
in toni complessi

# Suoni complessi

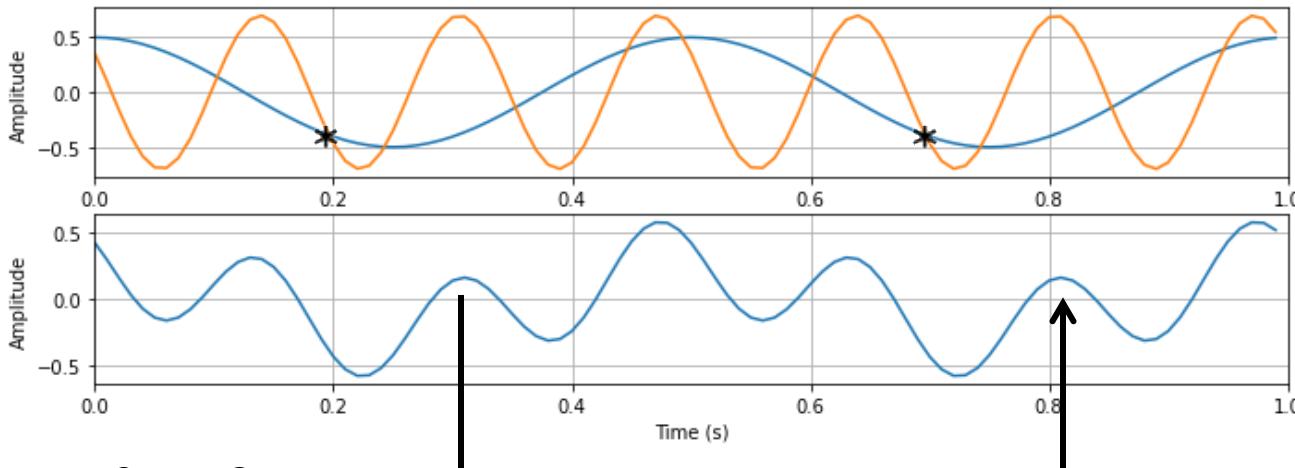
---

- Suoni complessi sono costituiti da più componenti (toni) con frequenze / ampiezze / fasi differenti
- Possiamo distinguerli in suoni complessi **periodici** o **a-periodici**
  - ✓ Dei suoni complessi periodici è possibile determinarne il periodo  $T$  e di conseguenza l' altezza del suono  $F_0 = 1/T$
  - ✓  **$F_0$**  NON è detta frequenza del suono ma FREQUENZA FONDAMENTALE
- Per creare suoni complessi periodici occorre sommare (solo e soltanto) componenti (toni) con  $f_k = k * F_0$  dove con  $k$  INTERO
  - ✓  $F_0$  è quindi il m.c.m. delle frequenze dei toni presenti nel segnale

# Suono complesso periodico

- Le componenti in frequenza (toni) dei suoni complessi periodici sono dette **armoniche**

- ✓ 1° armonica:  $F_0$
- ✓ k-esima armonica:  $f_k = k * F_0$



$$F_0 = 2$$

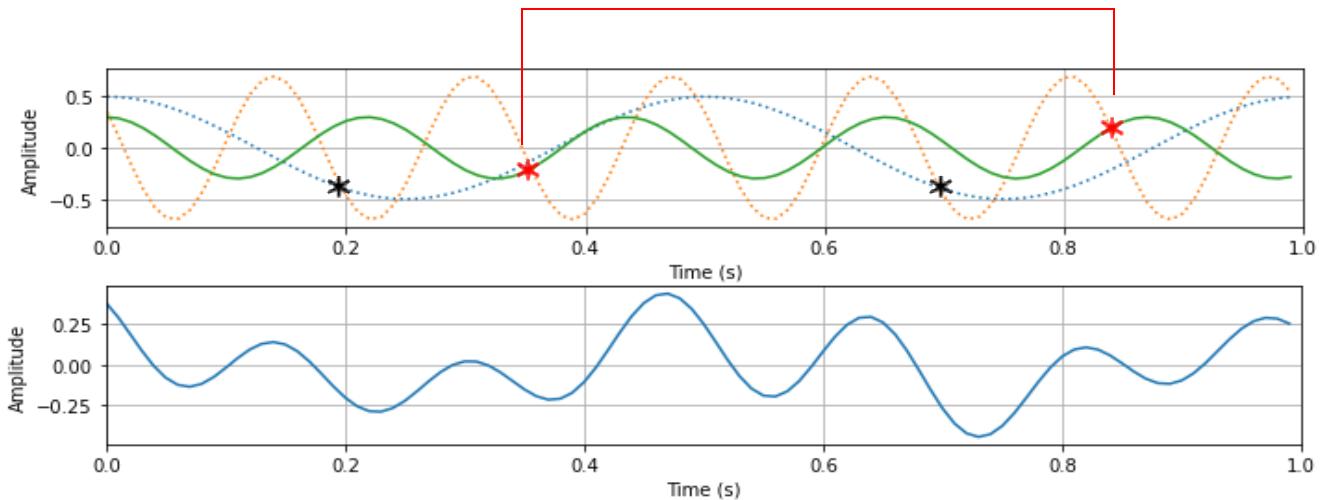
$$w1 = .5 * \text{np.cos}(2 * \text{np.pi} * F_0 * t + 0)$$

$$w2 = .7 * \text{np.cos}(2 * \text{np.pi} * 3 * F_0 * t + \text{np.pi}/3)$$

$$w = (w1 + w2)/2$$

# Suono complesso a-periodico

- Se si somma una componente non multipla intera di  $F_0$  il suono diventa a-periodico
  - ✓ e.g.  $2.3 * F_0$
- La periodicità è distrutta
  - ✓ Intuitivamente i picchi delle sinusoidi si spostano tra un periodo e l'altro



```
w1 = .5 * np.cos(2 * np.pi * F0 * t + 0)
w2 = .7 * np.cos(2 * np.pi * 3*F0 * t + np.pi/3)
w3 = .3 * np.cos(2 * np.pi * 2.3*F0 * t)
w = (w1 + w2 + w3) / 3
```

# Ritardo temporale vs ritardo di fase

---

- X

# Ritardo temporale vs ritardo di fase

---

- X

# Effetto della fase

---

- Seno e coseno differiscono per la fase iniziale
  - ✓  $\cos(x) = \sin(x+\pi/2)$  e  $\sin(x) = \cos(x-\pi/2)$

# Sintesi tono puro

## ■ Come generare una sinusoida?

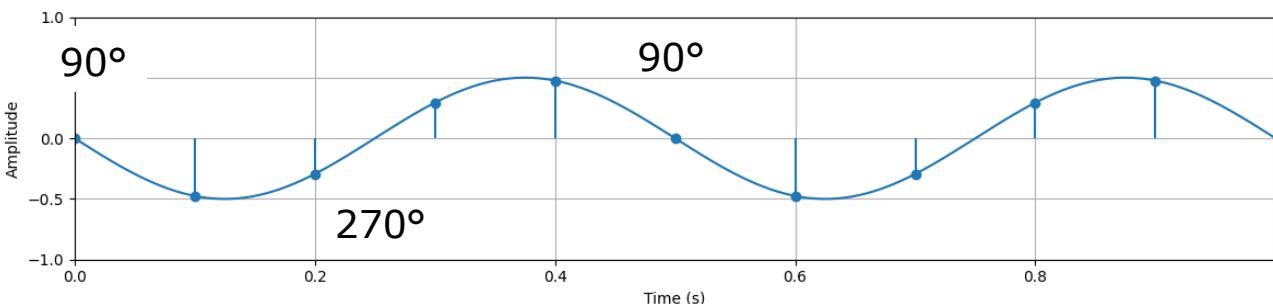
- ✓ Occorre generare una sequenza di valori campionando la sinusoida a intervalli  $\Delta t = 1/F_s$  (cioè calcolando i valori del coseno in quei punti)
- ✓ Genero quindi l'asse temporale (gli istanti di campionamento)

$$\hat{t} = i \cdot 1/F_s \quad \text{con } i=0..N-1 \text{ (secondi)}$$

- ✓ Calcolo la funzione discreta  $\hat{w} = A \cdot \cos(2\pi \cdot f \cdot \hat{t} + \varphi) = A \cdot \cos(\theta)$

$f: 1/s$  (Hz)  
 $t: s$   
 $\varphi: rad$   
 $\theta: rad$

- ✓ Ricordiamo che l'argomento del coseno è un angolo (in radianti) che, se la fase iniziale è 0, parte da 0 e avanza più o meno velocemente in funzione del fattore moltiplicativo frequenza  $f$ , modulo 360 o  $2\pi$



Se  $\cos$  allora  $\varphi = \pi/2$