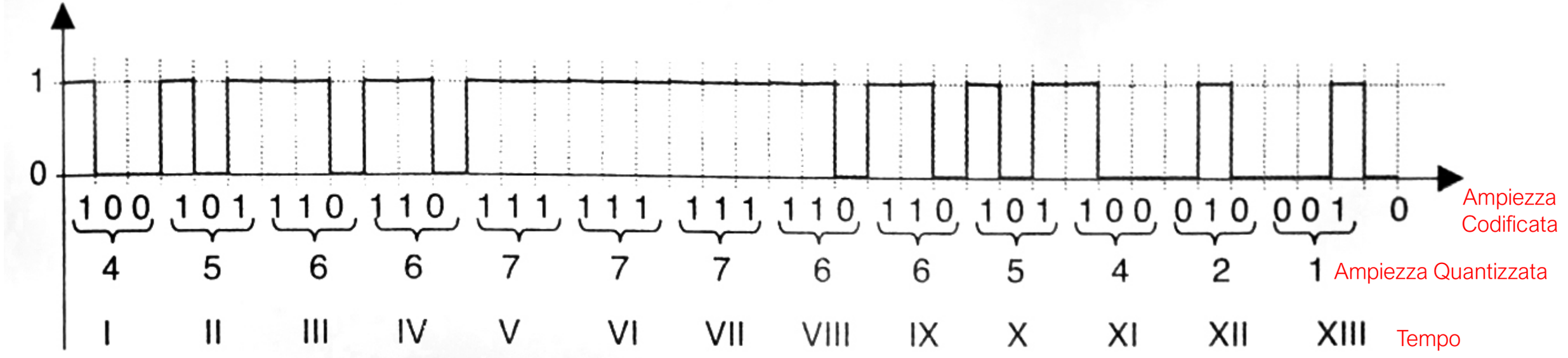
Codifiche e compressione

Audio Digitale – Codifica PCM

La **Pulse Code Modulation** (PCM), è forse la più semplice tecnica di codifica di un audio digitale. In effetti non si fa altro che considerare ogni singolo campione come un impulso e associarvi una parola binaria che ne rappresenta l'ampiezza. La lunghezza delle parole binarie dipende ovviamente dai bit di quantizzazione (lineare) utilizzati.



Nell'esempio si può osservare la codifica PCM a 3 bit di un segnale audio. I 13 campioni assumono valori tra 0 e 7.

Codifiche del segnale (dal testo)

- La differenza fondamentale dal punto di vista della rappresentazione binaria è se la codifica è con segno o senza segno

ID Quanto	Offset binary	Comp. a 2	Segno e Magnitudo
7	111_2 (7_{10})	011_2 ($+3_{10}$)	011_2 ($+3_{10}$)
6	110_2 (6_{10})	010_2 ($+2_{10}$)	010_2 ($+2_{10}$)
5	101_2 (5_{10})	001_2 ($+1_{10}$)	001_2 ($+1_{10}$)
4	100_2 (4_{10})	**000_2 ($+0_{10}$)**	**000_2 ($+0_{10}$)**
3	011_2 (3_{10})	111_2 (-1_{10})	**100_2 (-0_{10})**
2	010_2 (2_{10})	110_2 (-2_{10})	101_2 (-1_{10})
1	001_2 (1_{10})	101_2 (-3_{10})	110_2 (-2_{10})
0	**000_2 (0_{10})**	100_2 (-4_{10})	111_2 (-3_{10})

Compressione

Tecnica che mediante opportune strategie permette la riduzione della quantità di bit necessari a rappresentare in forma digitale una certa informazione.

Perché farlo?

1. Riduzione dello spazio di memoria occupato
2. Riduzione dei tempi (e costi) di trasmissione

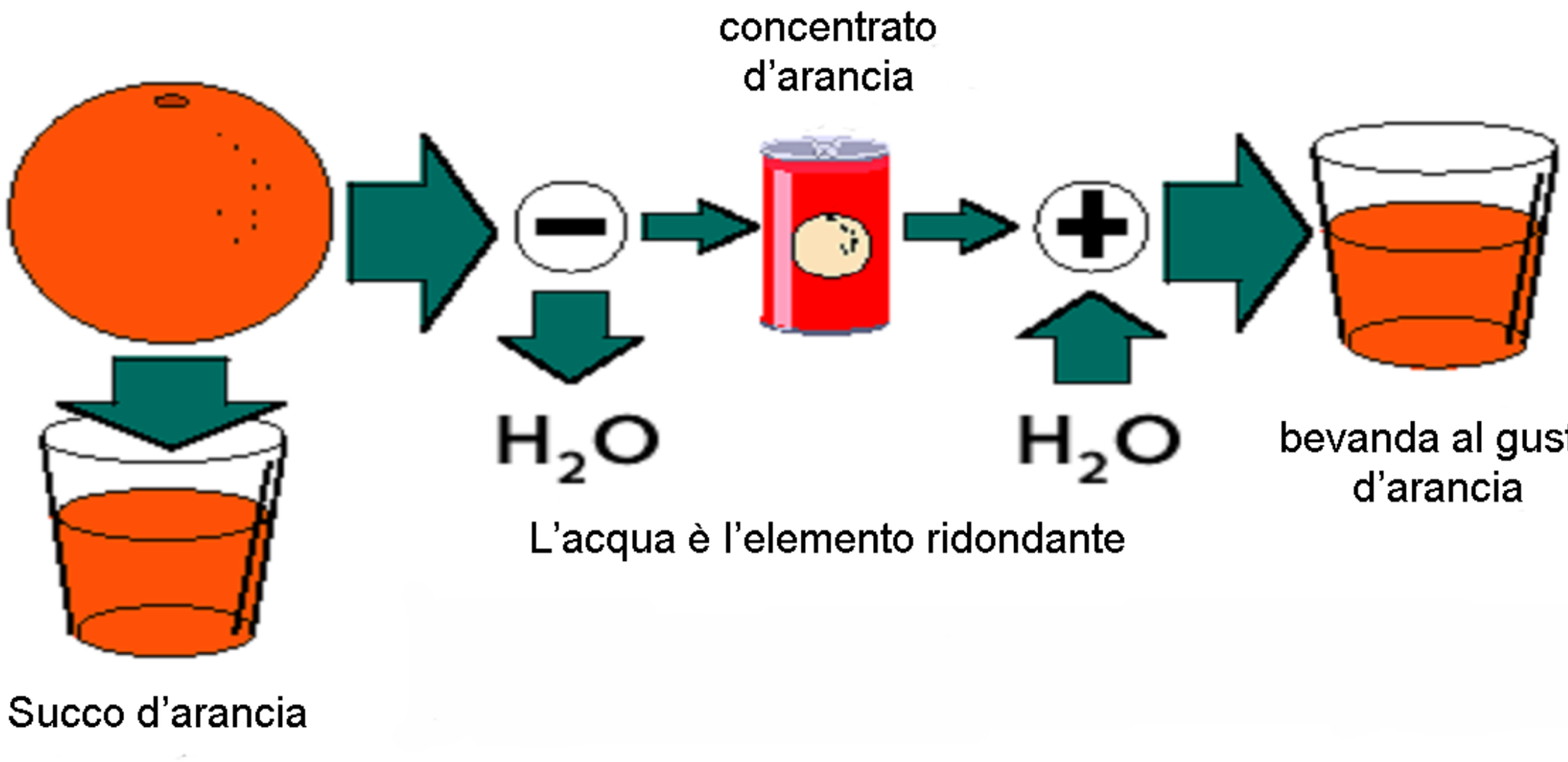
Come comprimere?

1. Riduzione della ridondanza nella codifica (legata a probabilità eventi)

2. Riduzione della ridondanza spaziale/temporale (legata a correlazione tra valori vicini)

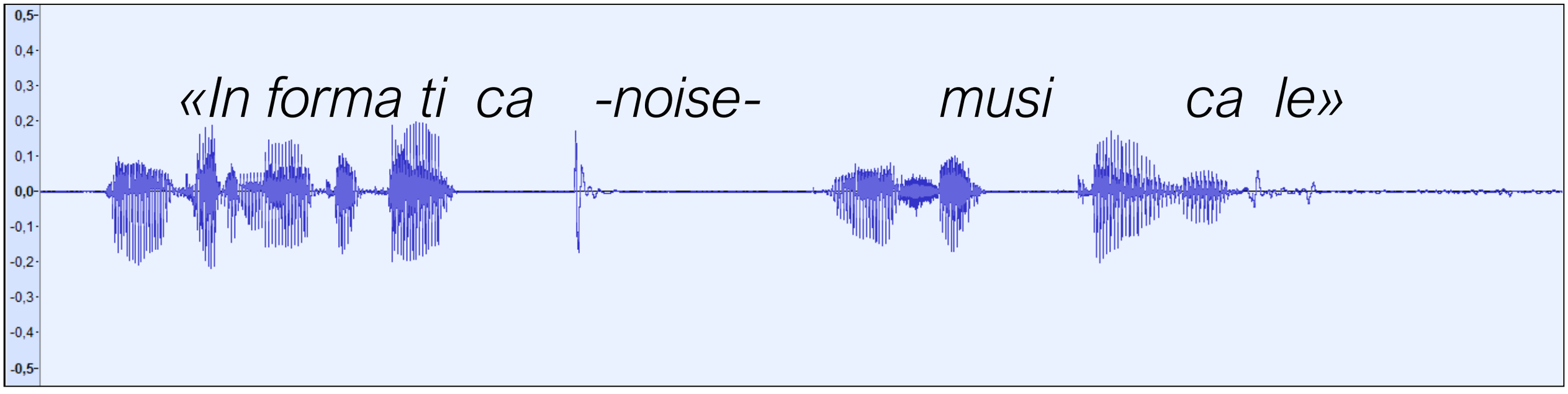
3. Riduzione dell'informazione irrilevante (legata spesso alla percezione)

Semplificando…



Compressione audio

- Come comprimere una traccia del genere?

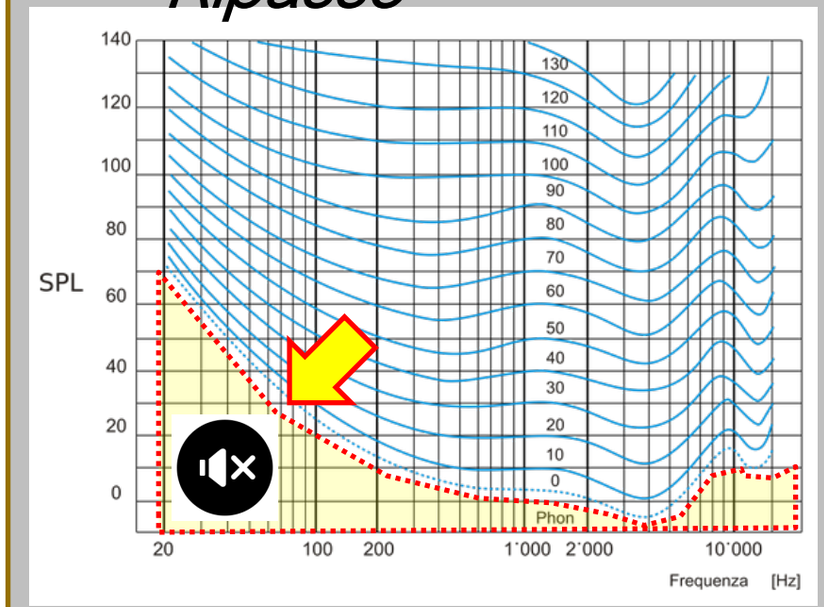


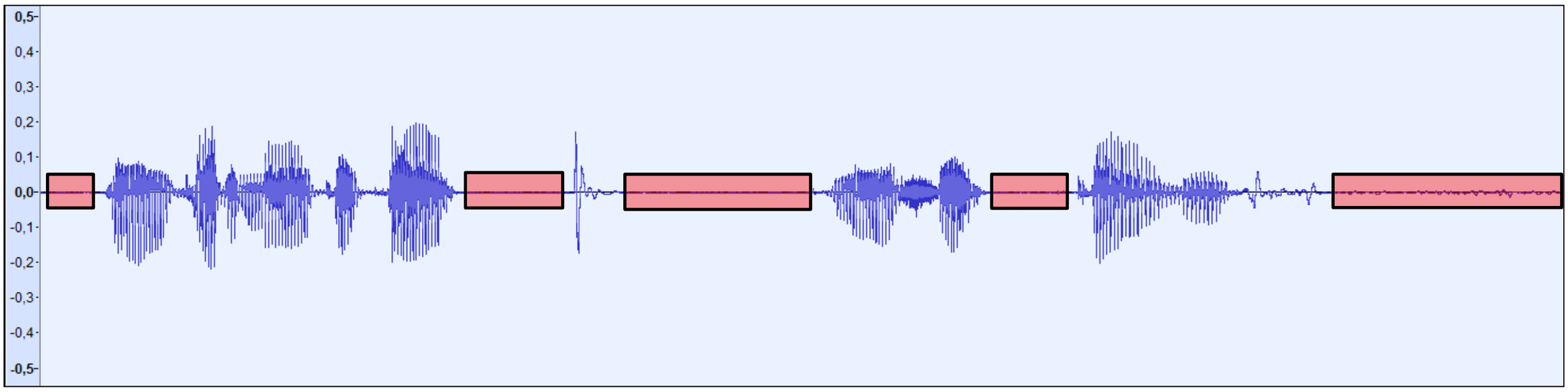
Compressione del silenzio (Metodo Naïve)

- Tecnica Lossy
  - Soglia di intensità sonora

Basata sulla soppressione di informazione irrilevante

Ripasso





Audio digitale – Spazio in memoria
Un esempio pratico

- CD Audio
  - Tasso di campionamento: 44,1kHz
  - Profondità in bit: PCM lineare 16 bit
    - Bitrate → 44,1kHz * 16 = 705,6kbps
  - Canali: Segnale stereo (2)
    - Bitrate → 705,6kbps * 2 = 1.411Mbps
    - 1 Minuto di registrazione: 44100*16*2*60/8 ~ 10MB

Il Bitrate è il «Tasso di trasferimento dati»

Audio Digitale – Codifica DPCM e ADPCM

La **Differential Pulse Code Modulation** (DPCM), è una versione della PCM pensata per comprimere in maniera **lossless**. Anziché codificare i valori di ampiezza, sia codificano solo le differenze. Si tratta di una semplice codifica **differenziale**.

La **Adaptive Differential Pulse Code Modulation** (ADPCM), è una tecnica di codifica più sofisticata della DPCM. Oltre a codificare le differenze utilizza un meccanismo di predizione unito ad un algoritmo di riquantizzazione che si «**adatta**» alle differenze da codificare. Brevemente diciamo che riquantizza le differenze più grandi tra valori reali e valori predetti (pertanto è **lossy**).

Differencing in DPCM

Il Differencing assume importanza se utilizzato in combinazione con la Run Length Encoding (RLE), in cui si codificano le lunghezze delle sequenze consecutive di simboli uguali (dette appunto Run, o Burst)

- *Differencing*:

…	100	101	102	103	103	103	102	101	…
…	…	+1	+1	+1	0	0	-1	-1	…

RLE → …; (3,+1); (2,0); (2,-1); …

LUT → …; 1; 2; 3

LUT	
(3,+1)	1
(2,0)	2
(2,-1)	3
…	…

Potremmo anche decidere di non adottare la RLE, e codificare le differenze in altro modo.

- In caso di differenze ridotte c'è un vantaggio effettivo nel codificare le differenze piuttosto che le codeword stesse
  - Si possono utilizzare delle LUT (Look-Up Table) (cioè delle tabelle-dizionario)

Nel caso del dizionario, inseriamo nella LUT le coppie calcolate nella RLE.

Predizione in ADPCM

- Per predire il campione successivo si somma ±1 al campione predetto attuale *(Predetto[n])*
  - Si sceglie la predizione più vicina al valore effettivo, poi si codifica la differenza rispetto al Valore Effettivo
  - Scelta diversa da ±1 ?

In questo esempio, in caso di dubbio (ambiguità) si sceglie sempre il valore predetto[n-1]+1

n	1	2	3	4	5	6	7	8	9	10	11	12	13	14
Valore Effettivo	100	101	102	103	103	103	102	101	100	100	101	101	100	97
Predetto[n-1]+1		101	102	103	104	105	104	103	102	101	102	103	102	101
Predetto[n-1]-1		99	100	101	102	103	102	101	100	99	100	101	100	99
Predetto[n]		101	102	103	104	103	102	101	100	101	102	101	100	99
Codifica		0	0	0	-1	0	0	0	0	-1	-1	0	0	-2

Audio Digitale – Codifiche A-law e μ-law

A-law e μ-law sono due algoritmi di codifica definiti nello standard G.711.

- Sono stati pensati per codificare digitalmente e trasmettere su una rete ISDN, la **voce al telefono**. Si parla quindi di suoni con frequenza da 0 a $4KHz$.

- Per ottimizzare l'utilizzo di banda, le due codifiche usano 8000 campioni al secondo $(4\,KHz \times 2)$ e una **quantizzazione non uniforme (logaritmica)** a 8 bit. Il bit rate sarà dunque di $64\,Kbps$ (8000×8), esattamente la larghezza di banda di una rete ISDN.

- La quantizzazione non lineare assicura maggiore precisione alle ampiezze più basse, garantendo una buona qualità con soli 8 bit.

Audio Digitale – μ-law

La codifica $\boldsymbol{\mu}$**-law** è in uso in Nord America e Giappone. Grazie alla quantizzazione non lineare, permette di ottenere con soli 8 bit la stessa qualità (es: SQNR) che si otterrebbe con una quantizzazione lineare a 14 bit.

Sia $\boldsymbol{X}$ il valore originale di ampiezza **normalizzato** tra [-1,1], $\boldsymbol{\mu}$ un fattore pari a $255\,(\,2^8-1)$, allora il valore codificato $\boldsymbol{Y}$ normalizzato in [-1,1] si calcola:

$$\mathrm{Y} = \mathrm{sign}(X)\,\frac{\ln(1+\mu|X|)}{\ln(1+\mu)}$$

I valori di Y attorno allo zero (più piccoli in valore assoluto) sono quelli a cui saranno dedicati più bit.

Audio Digitale – μ-law

Per ottenere i valori y a 8 bit con segno in [-128, 127] , partendo da valori x a 16 bit con segno in [-32768, 32767] si potrebbero utilizzare le seguenti normalizzazioni:

$$X = \frac{x + 32768}{32767.5} - 1$$

$$y = \text{round}(Y * 127.5)$$

In questo modo X sarà in [-1,1] e potrà essere applicata la formula vista nella slide precedente. Infine, partendo da Y compreso in [-1,1] si ottiene y nel range voluto che verrà codificato.

Nulla comunque vieta di utilizzare altre strategie. L'importante è che a valori di ampiezza più piccoli nel range originale, corrispondano più codeword tra le 256 disponibili.

Ovviamente l'audio così rappresentato dovrà essere decodificato per poter essere riprodotto.

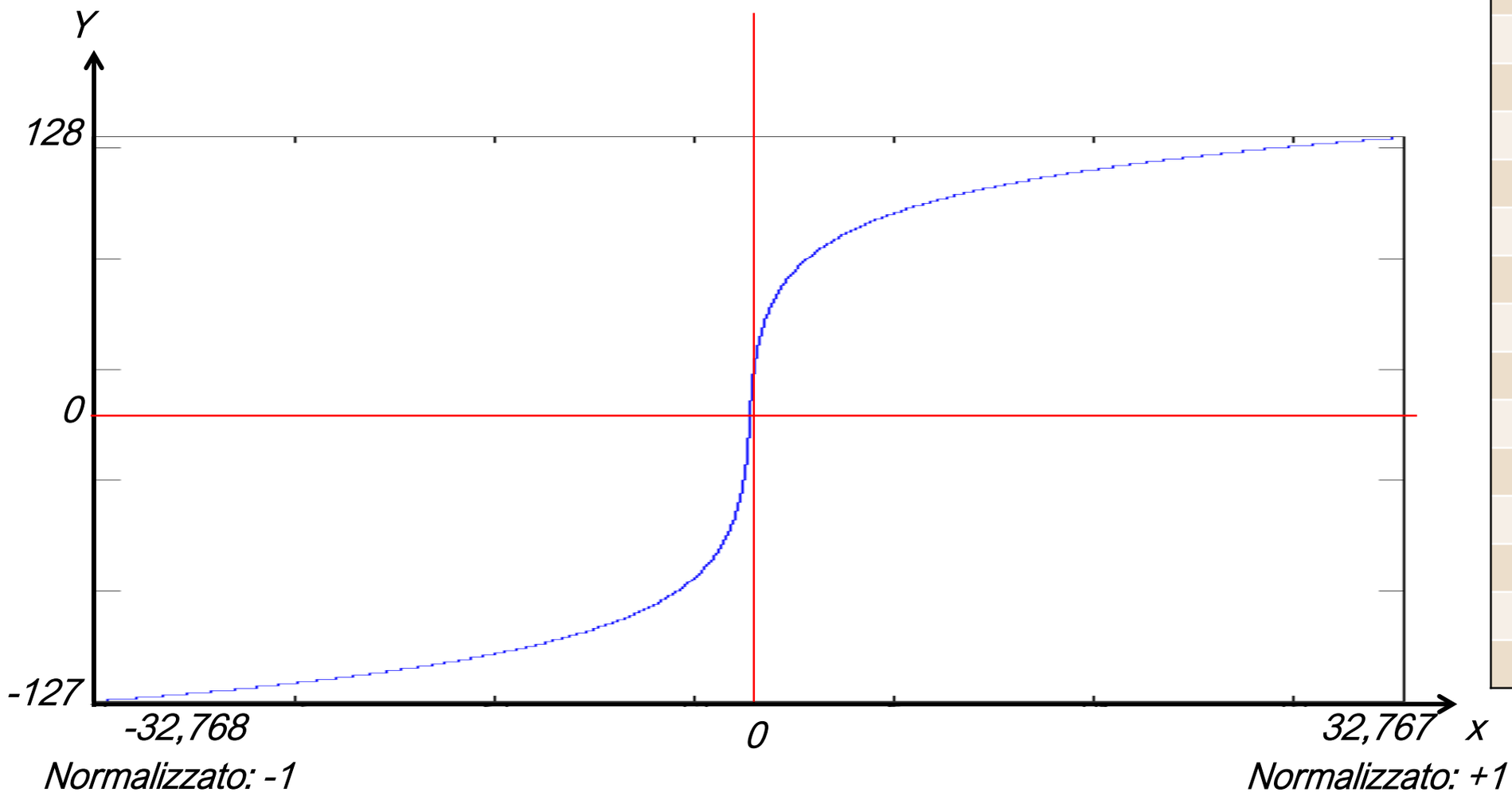
Audio digitale - μ-law, decodifica

Sia $\boldsymbol{Y}$ il valore di ampiezza codificato, **normalizzato** tra [-1,1], e $\boldsymbol{\mu}$ un fattore pari a $255\,(\,2^8-1)$, allora il valore decodificato $\boldsymbol{X}$ normalizzato in [-1,1] si può riottenere dalla seguente legge:

$$\mathrm{X} = \mathrm{sign}(\mathrm{Y})\,\frac{e^{\ln(1+\mu)|\mathrm{Y}|}-1}{\mu}$$

Queste formule sono generali. Fissando alcuni valori si possono costruire direttamente le versioni che permettono di passare da un intervallo di interi a 16 bit (es: [0, 65535]) ad uno ad 8 bit (es: [0, 255] oppure [-127,128]).

Esempio μ-law



Campione originale	Nuovo campione
-32768	-128
…	…
-32100	-128
-32000	-127
…	…
-200	-22
-100	-15
…	…
0	0
…	…
100	14
200	21
…	…
32000	126
32100	127
…	…
32767	127

Sull'asse orizzontale sono presenti valori a 16 bit (interi tra -32768 e 32767). Sull'asse verticale si trovano i corrispondenti interi a 8 bit (tra -128 e 127) ottenuti con la codifica μ-law. Si noti come i valori agli estremi siano quantizzati in maniera meno precisa.

Audio Digitale – A-law

La codifica **A-law** è in uso in Europa. Grazie alla quantizzazione non lineare, permette di ottenere con soli 8 bit la stessa qualità (es: SQNR) che si otterrebbe con una quantizzazione lineare a 13 bit.

Sia X il valore originale di ampiezza **normalizzato** tra [-1,1], A un fattore pari a 87.7 (o 87.6), allora il valore codificato Y normalizzato in [-1,1] si calcola:

$$\mathrm{Y} = \mathrm{sign}(X) \begin{cases} \frac{A|X|}{1+\ln A} & |X| \leq \frac{1}{A} \\ \frac{1+\ln A|X|}{1+\ln A} & \frac{1}{A} < |X| \leq 1 \end{cases}$$

Audio Digitale – A-law decodifica

Sia $\boldsymbol{Y}$ il valore codificato di ampiezza **normalizzato** tra [-1,1], $\boldsymbol{A}$ un fattore pari a 87.7(o 87.6), allora il valore decodificato $\boldsymbol{X}$ normalizzato in [-1,1] si può riottenere dalla seguente legge:

$$X = \text{sign}(Y) \begin{cases} \dfrac{|Y|(1+\ln A)}{A} & |Y| \leq \dfrac{1}{1+\ln A} \\ \dfrac{e^{|Y|(1+\ln A)-1}}{A} & \dfrac{1}{1+\ln A} < |Y| \leq 1 \end{cases}$$

Tutte le considerazioni sulla normalizzazione fatte per la codifica μ-law, valgono pure per A-law. Per entrambe le codifiche possono essere definite delle tabelle di conversione per passare da codeword di 14 a 8 bit (μ-law) e da 13 bit a 8 bit (A-law).

Floating Point Quantization

Si aggiungono dei bit extra a una codifica ottenuta da una quantizzazione lineare (uniforme). I bit extra fungono da traslatori della codifica.

Esempio con 1 bit extra e quantizzazione uniforme a 8 bit. Codifica finale a 9 bit.

- Se il bit extra è 0 codifico la sequenza originale a 8 bit aggiungendo uno 0 a sinistra (come bit più significativo)
- Se il bit extra è 1 aggiungo invece lo 0 a destra (meno significativo).

10110111 con bit extra 0 diventa -> **<u>0</u>**10110111
10110111 con bit extra 1 diventa -> 10110111**<u>0</u>**

- Divido intervallo da quantizzare (es: $2V$) nei range (*a*) $[-V/2, +V/2]$ e (*b*) $[-V, -V/2] \cup [V/2, V]$.
- Quando quantizzo, se il valore si trova nel range (*a*) quantizzo uniformemente e pongo bit extra a 0. Altrimenti, se il valore si trova nel range (*b*) divido per 2 il valore per ricondurmi al range (*a*), ma pongo bit extra 1.
- In questo modo le ampiezze più grandi in valore assoluto (range (*b*)) vengono codificate con passi di quantizzazione grandi il doppio rispetto a quelli nel più piccole (range (*a*)). **Perché?** Le codeword dei valori del range (*b*) finiranno sempre con **0 (solo pari).**

Floating Point Quantization

Per cui:

- Quindi con soli 8 bit e 1 bit extra posso codificare una gamma ottenibile con 9 bit, cioè codifico valori interi con segno tra -255 a 256 con soli 8 bit.

- Con 2 bit posso operare 4 diversi tipi di shift. Con 3 bit, 8 shift.

- In generale, con N bit di quantizzazione uniforme e P bit extra per lo shift si raggiunge un SNR in dB pari a circa $6N + 6P$ contro l'SQNR pari a circa $6N$ ottenibile con solo N bit per una quantizzazione uniforme.

- Si realizza spesso con numeri floating point in cui i bit extra fungono da **esponente** e quelli di quantizzazione uniforme da **mantissa**.

Fattori di compressione per codifiche basate su PCM

- Dipendono dalla implementazione della PCM:
  - IMA ADPCM: **4 a 1 (75%)**
  - ACE/MACE ADPCM: **2 a 1 (50%)**

Queste codifiche con compressione sono di tipo *lossy*

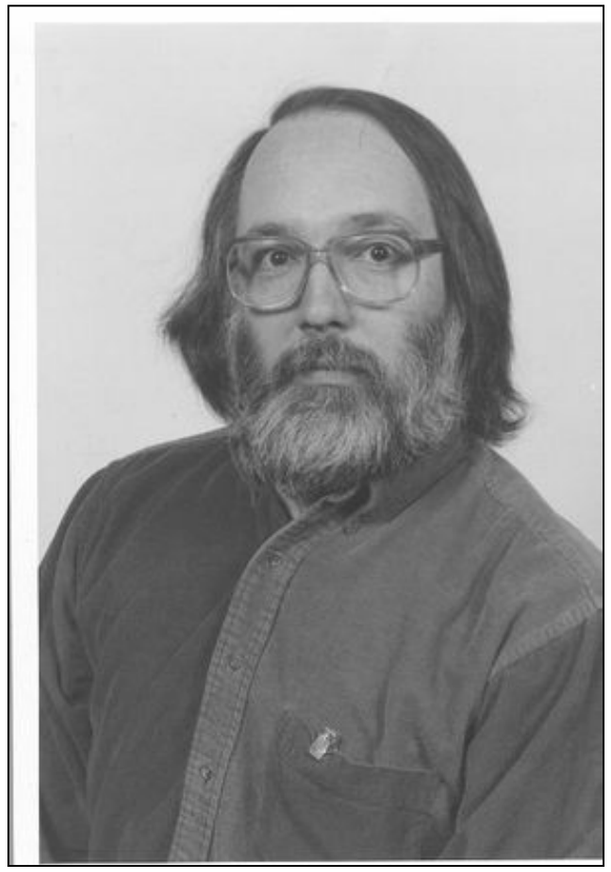
IMA: Interactive Multimedia Association, usato in MS Windows

ACE/MACE è la compressione APPLE

- Ma soprattutto dipende dall'utente, che stabilisce in base alle necessità la fedeltà vs compressione del segnale

James D. Johnston (?? – ancora in vita?)

- Noto come *Il padre delle codifiche di compressione di tipo percettivo*
- Responsabile di numerose codifiche all'interno dei formati MP3 e MPEG-2. Lavorò per 26 anni nei Bell Labs. Dopo aver lasciato i Bell Labs si è trasferito alla Microsoft.



Compressione percettiva (Entropia percettiva)

- J.D.Johnston ha fissato un limite teorico alla comprimibilità di un **segnale audio** se si vuole ottenere una codifica trasparente
  - **Codifica trasparente:** è una codifica (*lossy*) che produce un segnale alterato e codificabile con meno bit rispetto a quelli necessari per l'originale, ma non distinguibile dal segnale originale

Studio della *Entropia Percettiva*

- Tale limite è di circa <u>2.1 bit / campione</u>

Compressione percettiva e Codifica trasparente – Esempio

- CD Audio
  - Tasso di campionamento: 44,1kHz
  - PCM lineare 16 bit: 44.1kHz * 16 = 705.6kbps
  - Compressione a 64kbps
    - E' una codifica compressa trasparente?
  - Ogni campione verrà campionato con 64000 / 44100 = 1,45 bit / campione
    - 1.45 < 2.1 →→→ **Codifica NON trasparente**

Il bit rate compresso è minore di quello non compresso (705,6kbps). Questo significa che comprimendo, ogni campione verrà codificato con meno bit di quelli iniziali…

…non più 16 bit per campione, ma 1,45 bit per campione

Codifica trasparente

- La trasparenza non è una proprietà necessaria delle codifiche di compressione
- E’ più che altro una conseguenza diretta del bit-rate di compressione scelto
- → Tipicamente non si può scegliere il bit-rate, perché è dettato dalla strumentazione
  - Fissato il bit-rate è però possibile dire se la codifica sarà trasparente

La tecnica **Compansion**

(Cap. 3.6 – Pag.130 – IV edizione)

- Compansion = Compression + Expansion (intesi proprio come operatori dinamici)
  - Compressione in fase di registrazione
  - Espansione in riproduzione
- → Utilizzata negli schemi di compressione di tipo percettivo
- Ideata dalla Dolby negli anni ’60-’70 per risolvere i problemi di SNR sui nastri magnetici

Dal testo: *il rumore, in particolar modo il fruscio delle cassette audio, non è più di ampiezza costante e indipendente dal segnale: ora è più forte quando il segnale è più forte, ed è più debole quando il segnale è più debole*

Compressione di tipo percettivo

- Negli schemi di compressione di tipo percettivo vengono impiegate numerose tecniche, combinate in vari modi
  - Abbiamo appena (visto la *Companding*
- a questa aggiungiamo:
  - la *quantizzazione non uniforme* (in **μ-law** e **A-law**)
  - la codifica differenziale *ADPCM*
  - e altre 4 tecniche principali… →

Prendiamo in considerazione anche le **debolezze dell'udito umano** :
- La THQ (Threshold to Quiet)
- Le Bande Critiche e il mascheramento

Compressione di tipo percettivo

- Si basa su 4 tecniche principali:

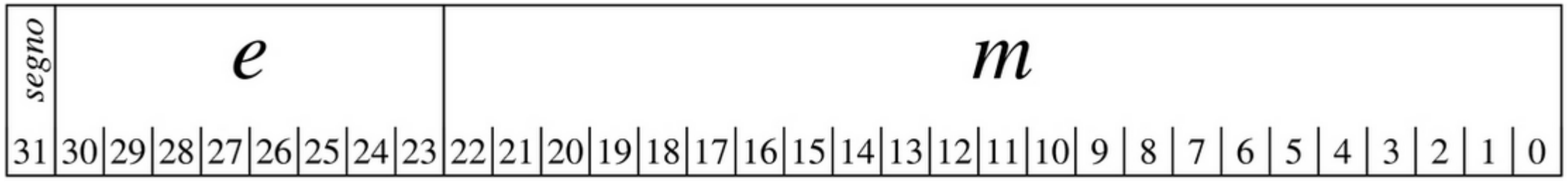
1. Block Coding
2. Transform Coding
3. Sub-band Coding
4. Huffman Coding

Block Coding

- Se uso Floating point quantization…
  - Esponente e mantissa

- … nelle tracce audio ci si aspetta che l'esponente vari pochissimo
  - Si può codificare l'esponente una volta sola per blocco

Es.: $+ \quad 10^{-e=-8} \times (m = 123456) = +0,00123456$



Block Coding - Pre-echi

- Problema dei pre-echi
  - Dovuto principalmente a impulsi alla fine di blocchi seguiti da blocchi che iniziano con bassa energia.

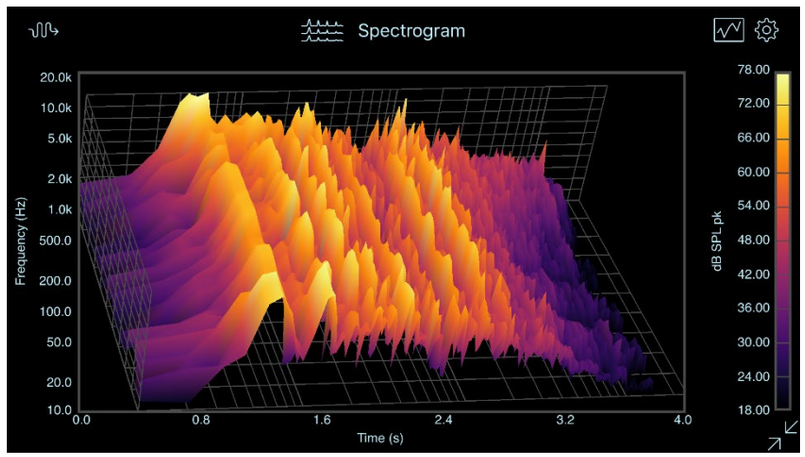
Introducono forti cambiamenti di scala (ordine di grandezza) → Rendono impossibile utilizzare un unico esponente per tutto il blocco

- Si può risolvere in 2 modi:
  - Ridurre la durata dei blocchi
  - Usare blocchi di durata variabile per circoscrivere i rumori impulsivi

In modo che sia più probabile che tutte le intensità dentro un blocco siano dello stesso ordine di grandezza

Transform Coding
Codifica nel dominio delle frequenze

- Il segnale audio nel dominio delle frequenze tende a variare meno rispetto al dominio dello spazio
  (→ *Spettrogramma*)



- Al posto della DFT applichiamo trasformate efficienti come la FFT o la DCT
  - La DCT è da preferire

FT: Fourier Transform
DFT: Discrete FT
FFT: Fast FT
DCT: Discrete Cosine Transform
MDCT: Modified DCT

Perché la DFT e la FFT utilizzano numeri complessi, mentre la DCT solo numeri reali, e le funzioni di base sono tutte (e solo) sinusoidi
[Pag 168 – Fig.4.12]

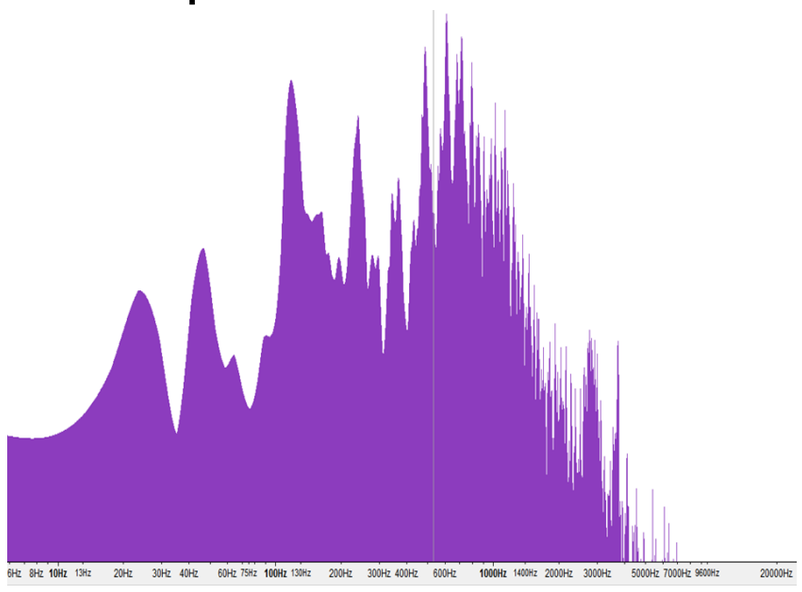
Transform Coding
Codifica nel dominio delle frequenze

- Vantaggiosa se applicata a blocchi con bassa gamma dinamica

- Per evitare i pre-echi
  - Si calcola la trasformata su intervalli sovrapposti per il 50%
    - Con questo metodo si ottiene però il doppio dei campioni necessari

Windowing
cioè sovrapposizione per finestre

- La compressione viene quindi applicata nel dominio delle frequenze,
  - Sui coefficienti delle trasformate
  - Sugli spettri (→ Sub-band Coding)



Spettro (continuo)

Sub-band Coding
Codifica per sottobande

- Analogamente alla codifica a blocchi →
- Divide lo spettro di frequenze in sottobande codificate in maniera individuale
  - Le sottobande con gamma dinamica ristretta possono essere codificate con meno bit
- Il processo di ***band-splitting*** non è semplice e richiede il giusto compromesso fra complessità di splitting e tasso di compressione

Ad esempio, si potrebbe applicare una suddivisione in base alle bande critiche (come nella MDCT)

Huffman Coding
Compressione di Huffman

- Codifica ottimale
  (→ si avvicina al limite di Shannon)
- Codici senza prefissi
- Compressione Lossless

- Algoritmo greedy:
  1. Selezione di due caratteri con frequenze minime
  2. Sostituzione dei due caratteri con uno fittizio la cui frequenza è la somma delle precedenti due
  3. Ripetere fino a ottenere il carattere con frequenza 1

Huffman Coding
Compressione di Huffman

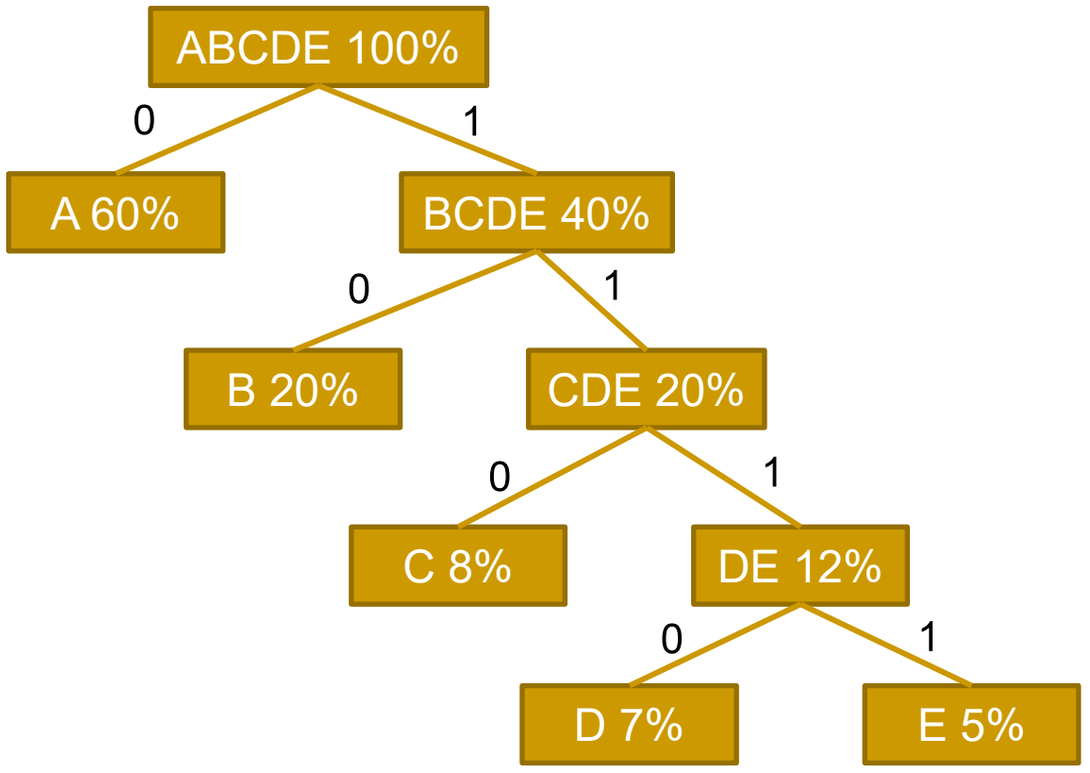
- A questo punto si ottiene un albero, la cui forma può variare in base alle scelte prese

- Si etichettano i rami binari con 0 e 1

- Si assegnano le codeword alle foglie leggendo dalla radice le etichette dei rami
  - Caratteri frequenti avranno codeword brevi
  - Caratteri rari avranno codeword lunghe

Huffman Coding
Esempio

Simboli e frequenze iniziali: A 60% B 20% C 8% D 7% E 5%



Simboli e codifiche finali:

A: 0, B: 10, C: 110, D: 1110, E: 1111

Schema generale di compressione di tipo percettivo

1. **Block-Coding**: Segmentazione della traccia audio in frame *quasi-stazionari* di 2-50 msec

2. **Transform Coding**: si passa all'analisi nel dominio delle frequenze o simili

3. **Sub-band Coding** opzionale, se si vuole ulteriormente ottimizzare la codifica del range dinamico

4. **Rimozione delle ridondanze** tramite codifiche lossy (ADPCM) o lossless (Huffman)

Codifica psicoacustica

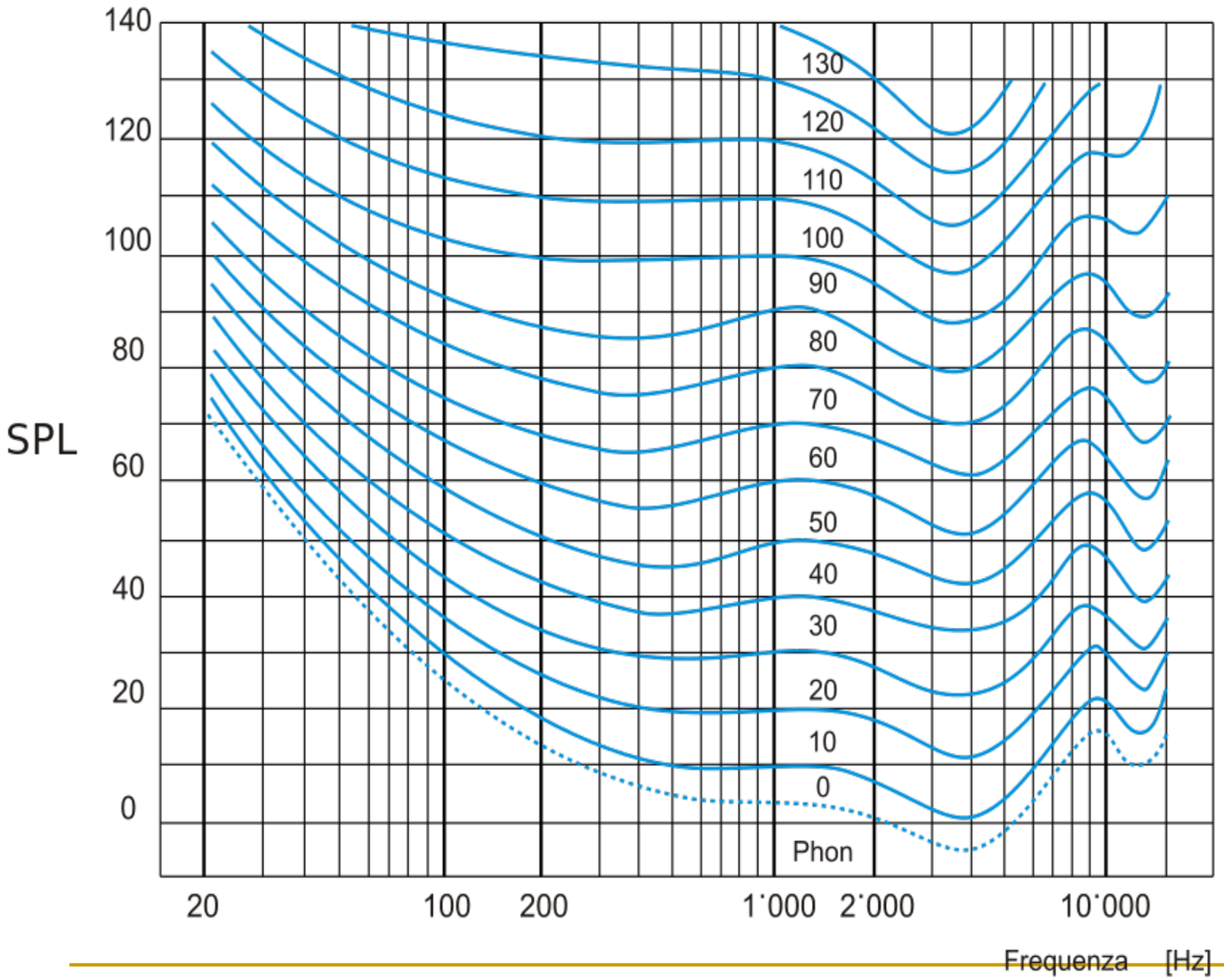
Come abbiamo visto in precedenza gli esseri umani percepiscono i suoni diversamente in base alla frequenza, l'ampiezza e anche al tempo tra due suoni consecutivi.

- L'idea è quella di utilizzare queste proprietà percettive degli esseri umani a proprio vantaggio;
- Cosa vuol dire? In pratica possiamo evitare di memorizzare (e riprodurre) tutto ciò che risulterebbe comunque impercettibile alla maggior parte degli esseri umani;

Le strategie di codifica basate su questi principi prendono il nome di **codifiche psicoacustiche**. La maggior parte delle codifiche avanzate sfrutta queste proprietà.

Codifica psicoacustica – Soglia udibilità

Sappiamo che toni con energia inferiore alla soglia di udibilità (per una certa frequenza) non sono percepiti. Possiamo sopprimerli.



Come descrivo la soglia che nel grafico delle curve isofoniche è rappresentata con una curva?

Codifica psicoacustica – Soglia udibilità

1. **Soluzione 1:** prendo la soglia minima tra tutte le soglie per ogni frequenza, ossia la soglia per il tono a 4Hz. Funziona, ma potrei fare di meglio.

2. **Soluzione 2:** approssimo la soglia al variare della frequenza f con la seguente equazione.

$$T_q(f) = 3{,}64\left(\frac{f}{1000}\right)^{-0.8} - 6{,}5\, e^{-0.6*\left(\frac{f}{1000}-3.3\right)^2} + 10^{-3}\left(\frac{f}{1000}\right)^4$$

Se il contributo energetico di una certa frequenza f è inferiore $T_q(f)$ allora lo sopprimo.

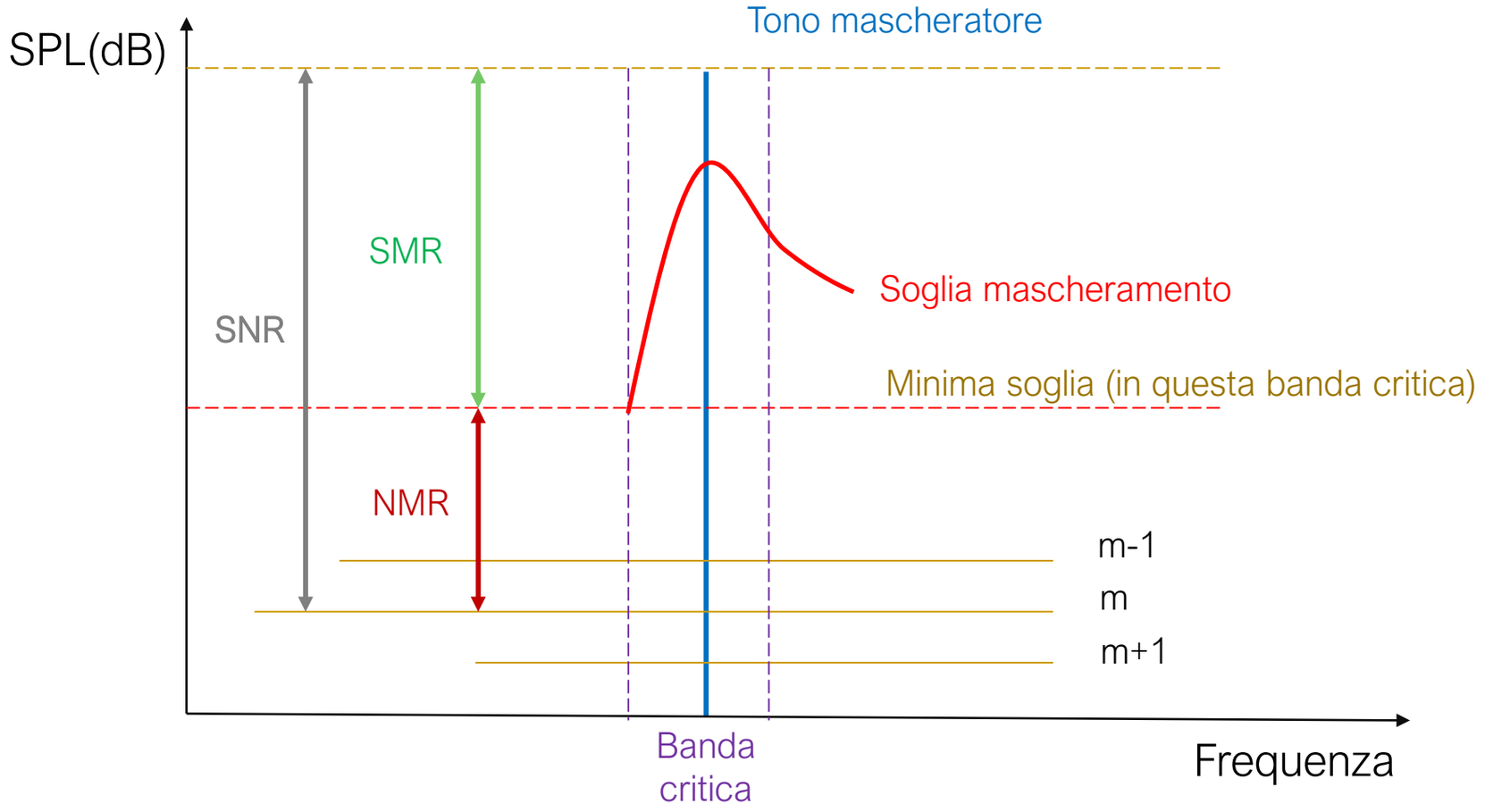
Codifica psicoacustica – Mascheramento

L'altra possibilità è sfruttare il fenomeno del mascheramento per scegliere il numero di bit di quantizzazione opportuno, tenendo presente che la distorsione (rumore) sotto la soglia di mascheramento di una certa banda critica non verrebbe percepito.

- Ricordiamo che il rapporto segnale rumore (SNR) può essere espresso, in forma logaritmica come $\log(segnale) - \log(rumore)$.
- Introduciamo anche il rapporto segnale maschera (SMR) e il rapporto rumore maschera (NMR).
- SMR è la distanza tra l'ampiezza del tono mascheratore e la minima soglia di mascheramento (nella banda critica analizzata)
- NMR è la distanza tra la distorsione introdotta quantizzando a m bit e la minima soglia di mascheramento.
- Per cui: SNR=SMR+NMR.

Codifica psicoacustica – Mascheramento

Quindi? Fintanto che il rumore rimane sotto la minima soglia di mascheramento posso diminuire il numero di bit, perché comunque tale rumore non sarebbe percepito. Mi basta che SNR sia non inferiore a SMR.



Codifiche per modelli

Le **codifiche per modelli** si basano sulla costruzione di un modello matematico che simuli una specifica sorgente sonora. Ciò che alla fine si rappresenterà sarà il comportamento della sorgente simulata.

Un' esempio tipico è quello della voce umana. Si costruisce un modello che descriva il suono in termini di vibrazioni di ogni corda vocale e di modulazioni dovute alla gola e alla bocca.

LPC (*Linear predictive coding*) è una codifica di questo tipo. Ciò che in effetti si memorizza è la differenza tra il valore predetto dal modello e quello reale.