FORMATI AUDIO PCM

SEGNALE DISCRETO

- Per segnale discreto (nel tempo) si intende una successione di valori di una certa grandezza dati in corrispondenza di una serie di valori discreti nel tempo
  - E' una funzione, o un segnale, con valori forniti in corrispondenza ad una serie di tempi scelti nel dominio dei numeri interi

- Ciascun valore della successione è chiamato campione

- Quando un segnale discreto è composto da una serie di valori ottenuti in corrispondenza di istanti spaziati uniformemente nel tempo, si dice che è associato ad una particolare frequenza di campionamento

PERIODO E FREQUENZA

- La frequenza è una grandezza che concerne fenomeni periodici o processi ripetitivi. Essa viene data dal numero di cicli completati in una data unità di tempo. Il periodo è l'intervallo di tempo in cui si completa un ciclo

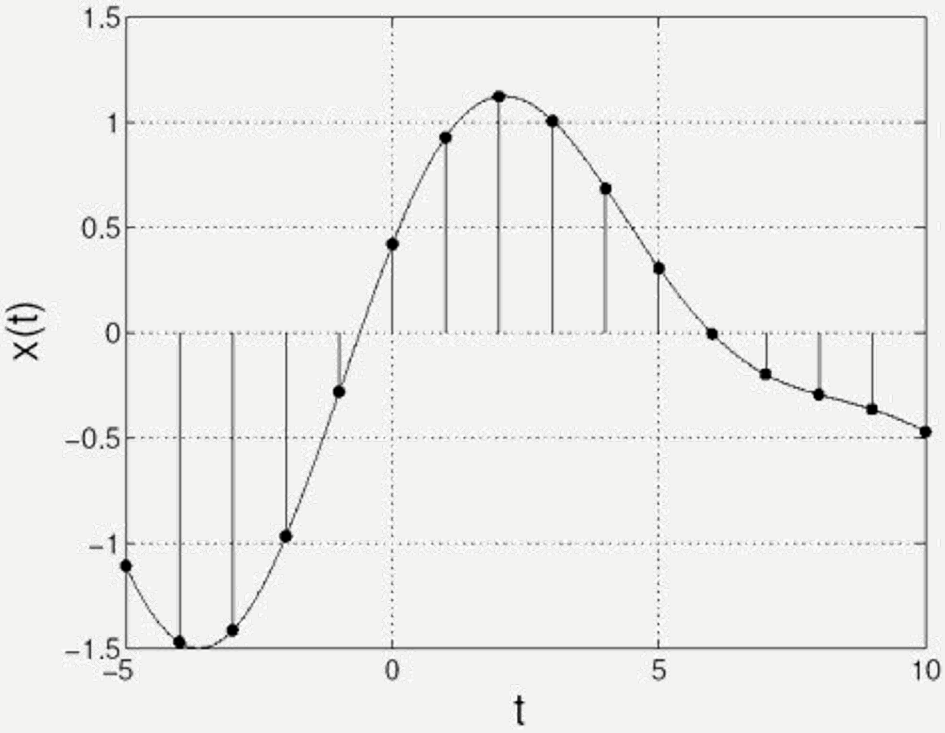
- Tali grandezze sono legate tra loro dal rapporto $f = 1 / T$ (proporzionalità inversa)

- Unità di misura per misurare T e f:
  - Periodo: secondi [s]
  - Frequenza: hertz [Hz]

- Esempio: se avvengono 71 iterazioni in 15 secondi
  $f = 71 / 15 = 4{,}7$ Hz; $T = 15 / 71 = 1 / 4{,}7 = 0{,}21$ s

CAMPIONAMENTO A FREQUENZA COSTANTE

- Un esempio di fenomeno ripetitivo nel tempo è la lettura periodica dell'ampiezza di una forma d'onda, effettuata a intervalli regolari (periodo di campionamento, frequenza di campionamento)



SEGNALE DIGITALE

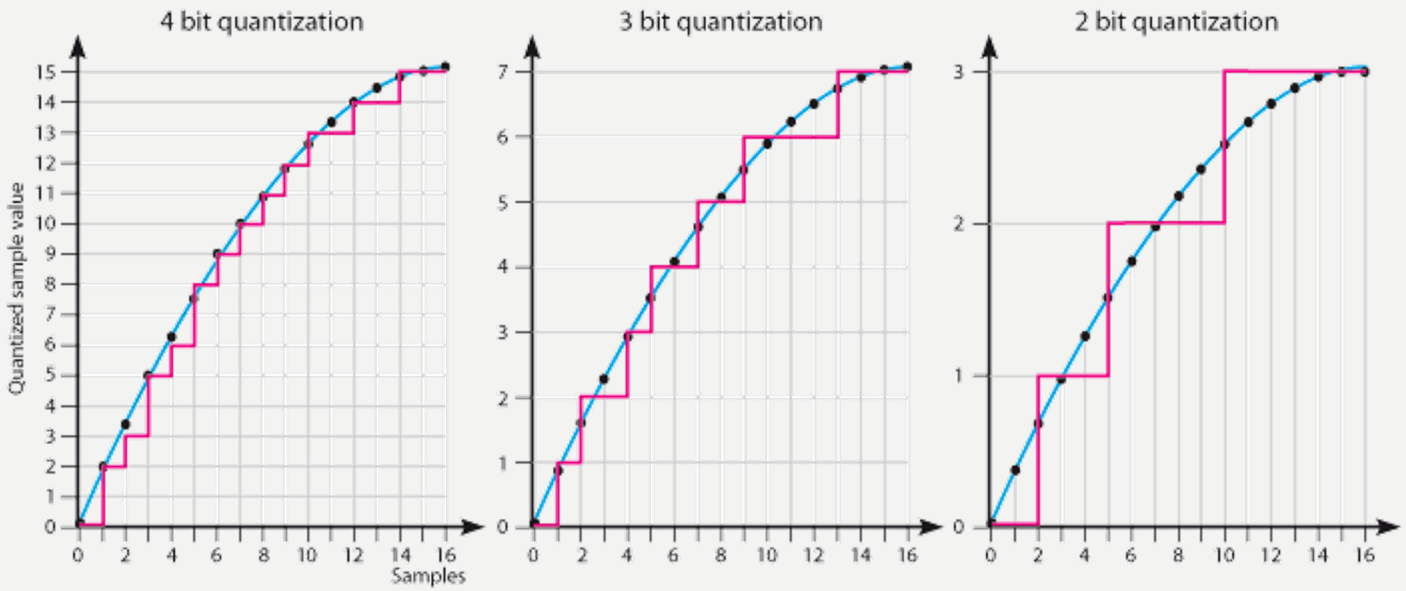
- Un segnale digitale è un segnale discreto che può assumere soltanto valori appartenenti ad un insieme discreto

- Il procedimento di conversione di un segnale discreto (segnale continuo campionato in valori di tempo discreti) in un segnale digitale è detto quantizzazione

- La quantizzazione provoca perdita di informazione, in quanto i valori effettivamente assunti dal segnale sono troncati o arrotondati

QUANTIZZAZIONE ED ERRORI DI QUANTIZZAZIONE

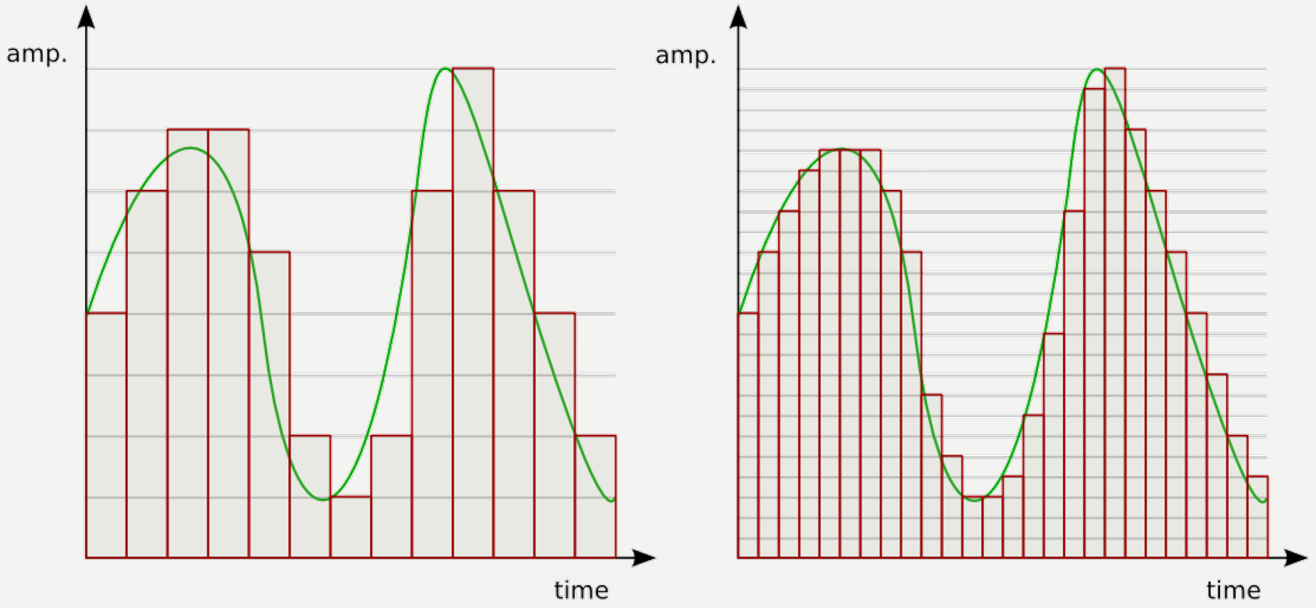
- Nel dominio digitale, la quantizzazione richiede di stabilire un numero n di bit su cui rappresentare il valore quantizzato, che potrà quindi assumere uno dei 2^n possibili valori ammessi



CONVERSIONE ANALOGICO-DIGITALE DELL'AUDIO

Nel caso di segnali audio, la conversione analogico-digitale si compone di 2 passaggi:

- Campionamento (discretizzazione sull'asse del tempo)
- Quantizzazione (discretizzazione sull'asse delle ampiezze)



TEOREMA DEL CAMPIONAMENTO DI NYQUIST-SHANNON

- Il teorema del campionamento è la base della teoria dei segnali: mette in relazione il contenuto informativo di un segnale campionato con la frequenza di campionamento

- Esso definisce la minima frequenza necessaria per campionare un segnale analogico senza perdere informazioni e per poter quindi ricostruire il segnale analogico tempo-continuo originario

- Tale frequenza è detta frequenza di Nyquist

TEOREMA DEL CAMPIONAMENTO DI NYQUIST-SHANNON

- In una conversione analogico-digitale la minima frequenza di campionamento necessaria per evitare ambiguità e perdita di informazione nella ricostruzione del segnale analogico originario (ovvero nella riconversione digitale-analogica) con larghezza di banda finita e nota deve essere maggiore del doppio della frequenza massima presente nel segnale originario

- Questo spiega perché la frequenza di campionamento nello standard CD-DA sia 44.100 Hz

PULSE CODE MODULATION (PCM)

- La pulse-code modulation («modulazione a codice di impulsi») è un metodo di rappresentazione digitale di un segnale analogico

- Nell'ambito dei segnali audio è possibile diagrammare sull'ascissa i tempi e sull'ordinata le ampiezze.
  Applicato a tale diagramma, il metodo utilizza un campionamento dell'ascissa del segnale a intervalli regolari; i valori letti vengono poi quantizzati in ordinata ed infine digitalizzati, ossia codificati in forma binaria

CARATTERISTICHE DEI FORMATI PCM

- La codifica PCM provvede a salvare i dati audio senza alcun tipo di compressione dati

- Conseguenze:
  - i file risultanti sono di elevate dimensioni, ma non richiedono elevata potenza di calcolo per essere riprodotti
  - la codifica è lossless, quindi viene spesso utilizzata dagli utenti professionali per memorizzare l'audio
  - questi formati sono ampiamente supportati dai software in uso, e in generale sono portabili tra diversi sistemi operativi

FORMATI WAV E AIFF

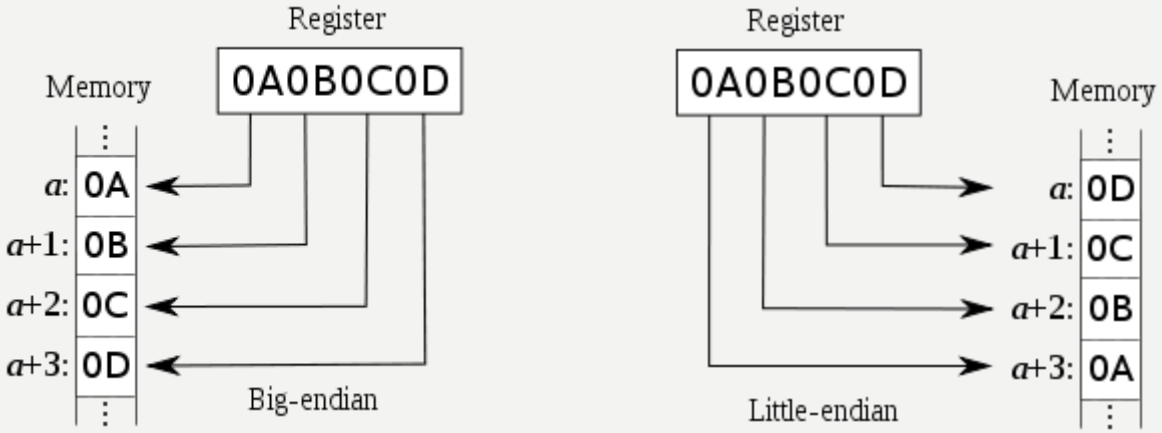
- WAV (o WAVE), contrazione di WAVEform audio file format (formato audio per la forma d'onda) è un formato audio di codifica digitale sviluppato da Microsoft e IBM per personal computer IBM compatibili

- È una variante del formato RIFF di memorizzazione dei dati: i dati vengono salvati in «chunk» (blocchi)

- Come approccio, WAV è simile al formato AIFF (Audio Interchange File Format) utilizzato dai computer Apple Macintosh, sviluppato dalla Apple Computer basandosi sull'Interchange File Format della Electronic Arts
  - I file in formato WAV o AIFF sono compatibili con i sistemi operativi Windows e Macintosh

CONFRONTO TRA WAV E AIFF

- I dati nel formato WAV vengono memorizzati con la notazione little endian, essendo progettato per computer con processori Intel o compatibili. Tradizionalmente gli AIFF adottano la notazione big endian, ma recentemente (con Mac OS X) Apple ha introdotto anche la codifica little endian



WAV E AIFF E CODIFICA PCM

- Essendo basati su standard per l'interscambio di informazioni, i formati supportano varie modalità di immagazzinamento dei dati

- Esiste una grande varietà di codec disponibili per i file .WAV:
  - PCM (il più diffuso)
  - ADPCM
  - GSM
  - CELP
  - SBC
  - TrueSpeech
  - MPEG Layer-3

- Nel corso delle lezioni si prenderà in considerazione il formato WAV con codifica PCM

TABELLA DI CONFRONTO DEI CODEC

Formato	Bitrate	1 Min
11 025 Hz 16 bit PCM	176.4 kBit/s	1292 kB
8 000 Hz 16 bit PCM	128 kBit/s	938 kB
11 025 Hz 8 bit PCM	88.2 kBit/s	646 kB
11 025 Hz µ-Law	88.2 kBit/s	646 kB
8 000 Hz 8 bit PCM	64 kBit/s	469 kB
8 000 Hz µ-Law	64 kBit/s	469 kB
11 025 Hz 4 bit ADPCM	44.1 kBit/s	323 kB
8 000 Hz 4bit ADPCM	32 kBit/s	234 kB
11 025 Hz GSM6.10	18 kBit/s	132 kB
8 000 Hz Mp3 16 k	16 kBit/s	117 kB
8 000 Hz GSM6.10	13 kBit/s	103 kB
8 000 Hz Lernout & Hauspie SBC 12 k	12.0 kBit/s	88 kB
8 000 Hz DSP Group Truespeech	9 kBit/s	66 kB
8 000 Hz Mp3 8 k	8 kBit/s	60 kB
8 000 Hz Lernout & Hauspie CELP	4.8 kBit/s	35 kB

FORMATO WAV-PCM CANONICO

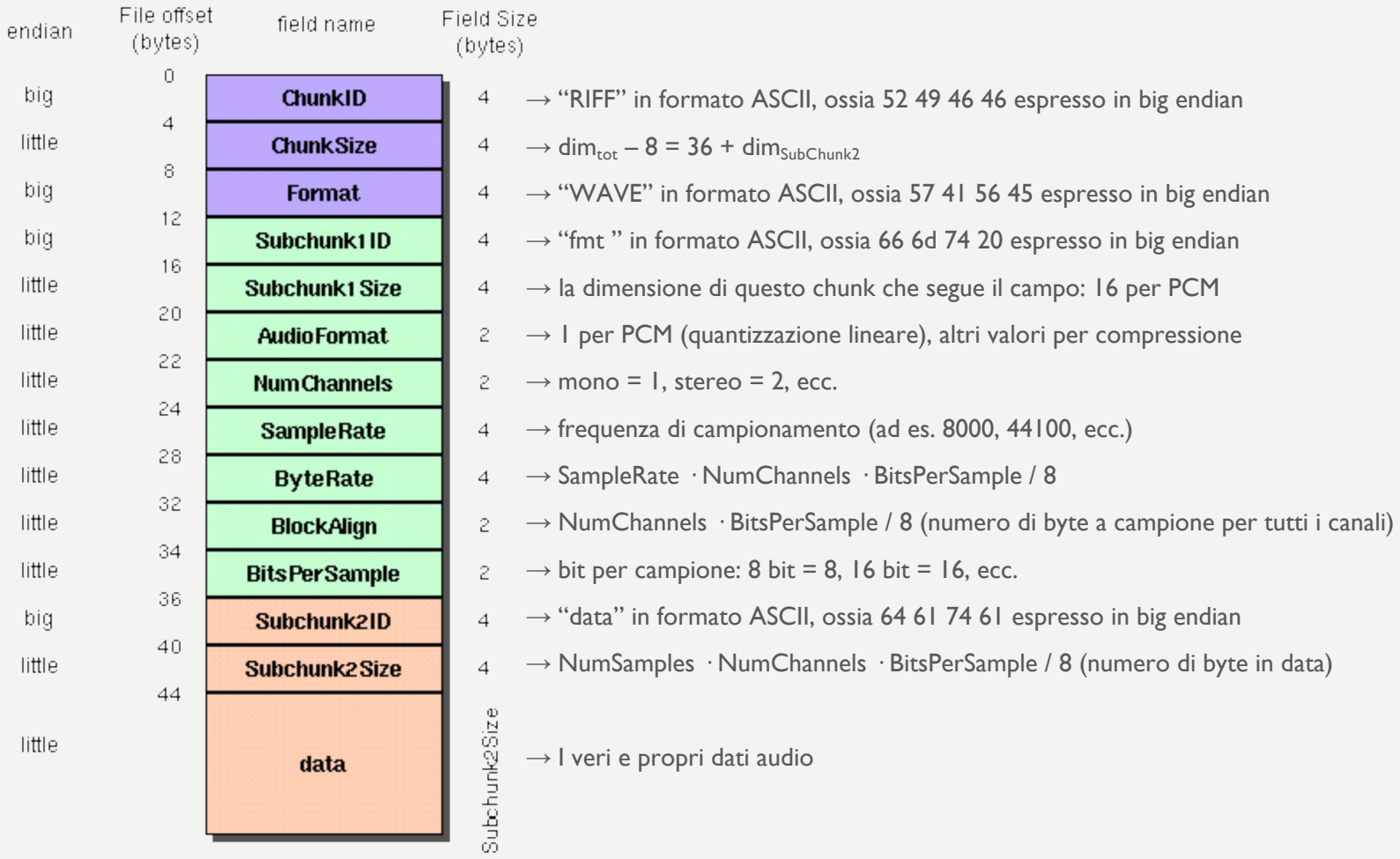
- Il formato WAV è un sottoinsieme delle specifiche Microsoft RIFF (Resource Interchange File Format) per la codifica di informazione multimediale

- I file RIFF sono costituiti da un header seguito da una sequenza di chunk, ossia strutture dati con un formato comune

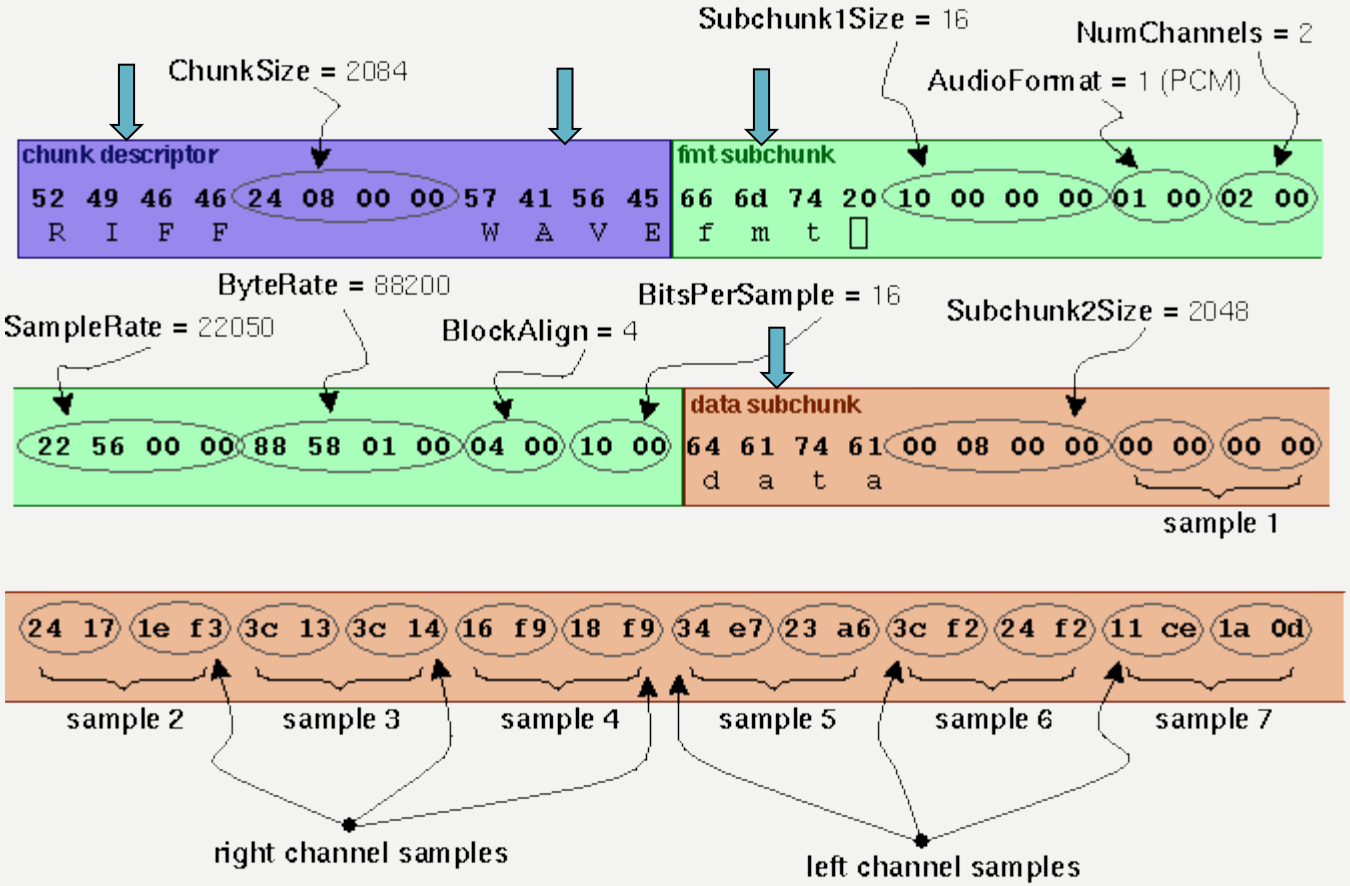
- Spesso un file WAV è un file RIFF con un singolo chunk "WAVE" che consta di due sotto-chunk:
  - Chunk fmt, che specifica il formato dei dati
  - Chunk data, che contiene i veri e propri dati

- Chiameremo questa strutturazione di un file WAV forma canonica

STRUTTURA DEL WAV-PCM CANONICO



ESEMPIO (PRIMI 72 BYTE)



Attenzione: solo gli ID (identificati da frecce azzurre) sono big endian, tutti gli altri casi sono little endian (quindi i byte, ossia le coppie di cifre esadecimali, vanno riordinati)

Esiste una versione completamente big endian di RIFF: il RIFX

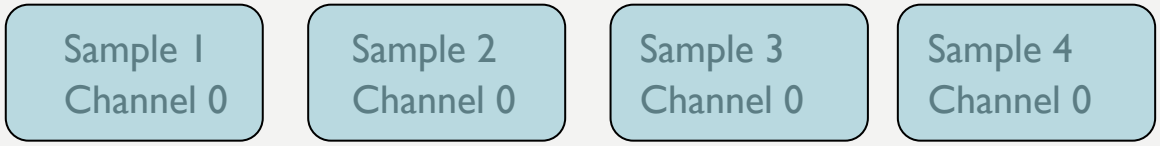
OSSERVAZIONI SUI CAMPIONI AUDIO

- Quantizzazione:
  - 8 bit a campione → i campioni vengono salvati come interi a 8 bit senza segno [0..255]
  - 16 bit a campione → notazione in complemento a 2 [-32768..32767]

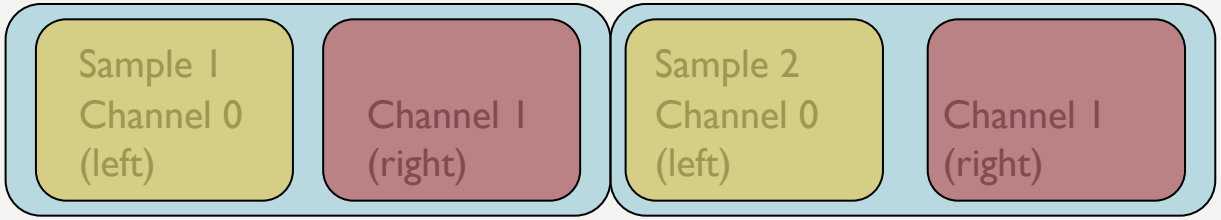
- Data packing:
  - Se il file è PCM mono a 8 bit per campione, il chunk dati si presenta come un array di byte in cui ogni elemento è un singolo campione
  - In che modo vengono rappresentati e interfoliati i dati dei campioni, quando i campioni per canale occupano più di 1 byte e sono presenti più canali?
    Per l'ordinamento dei byte che si riferiscono a un singolo campione, si ricordi che la notazione RIFF è little endian
    Per l'interfoliazione dei canali, si veda la slide successiva

ESEMPI DI DATA PACKING

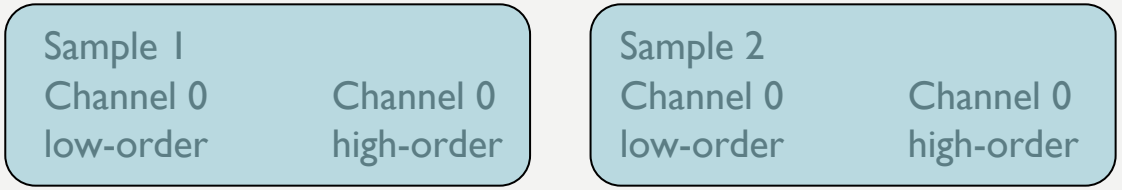
Mono, 8 bit



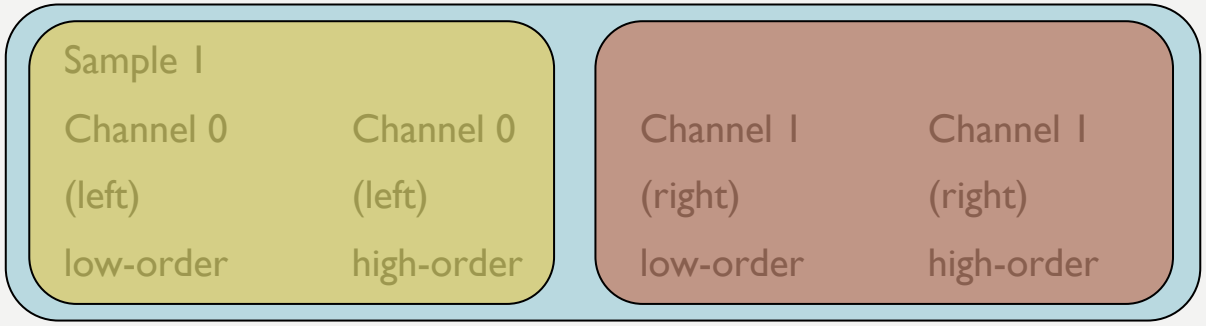
Stereo, 8 bit



Mono, 16 bit



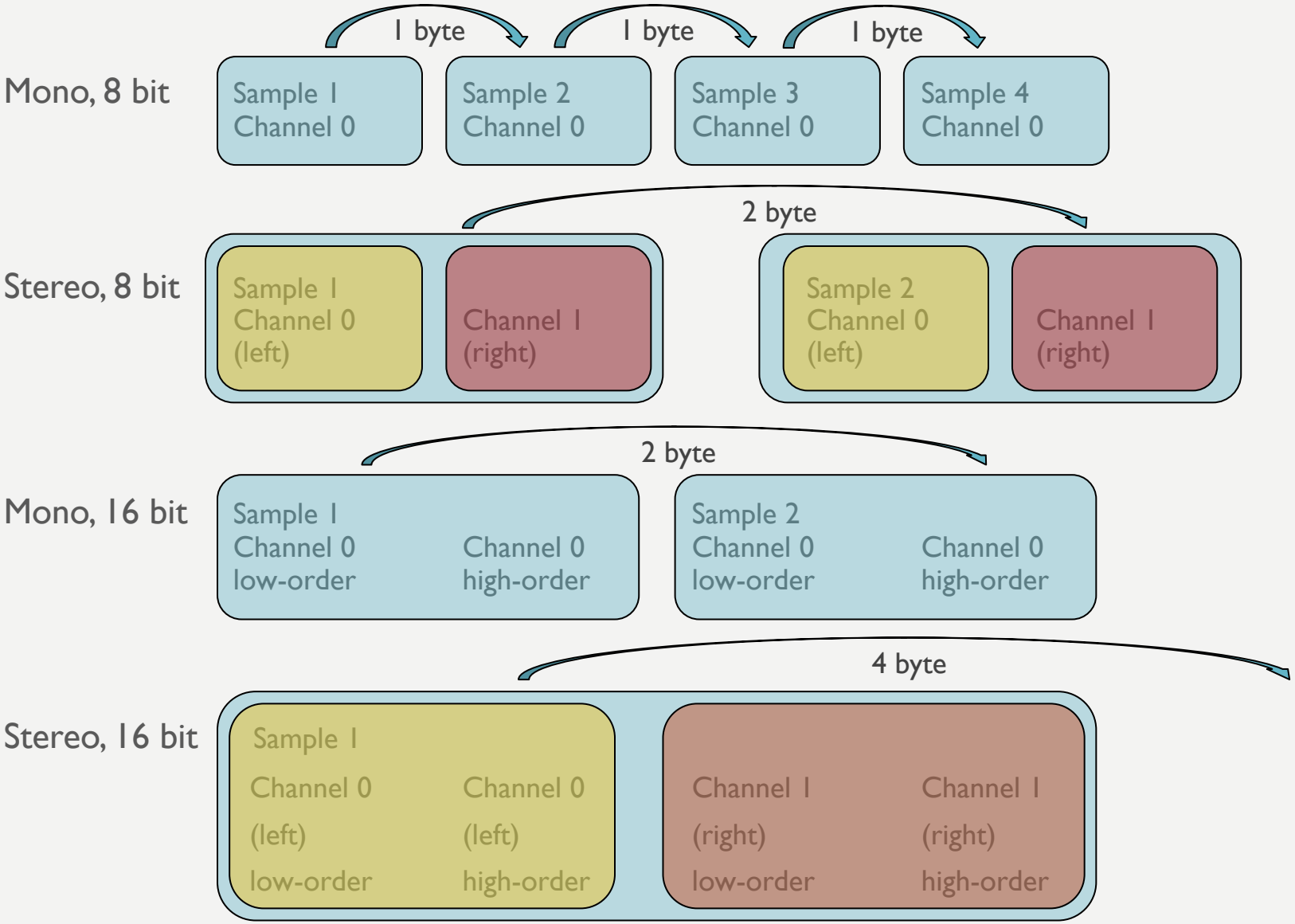
Stereo, 16 bit



I riquadri azzurri contornano le informazioni audio relative allo stesso istante di tempo

Block align = NumChannels · BitsPerSample / 8

E' lo spiazzamento in byte tra blocchi relativi alla descrizione audio di istanti temporali successivi

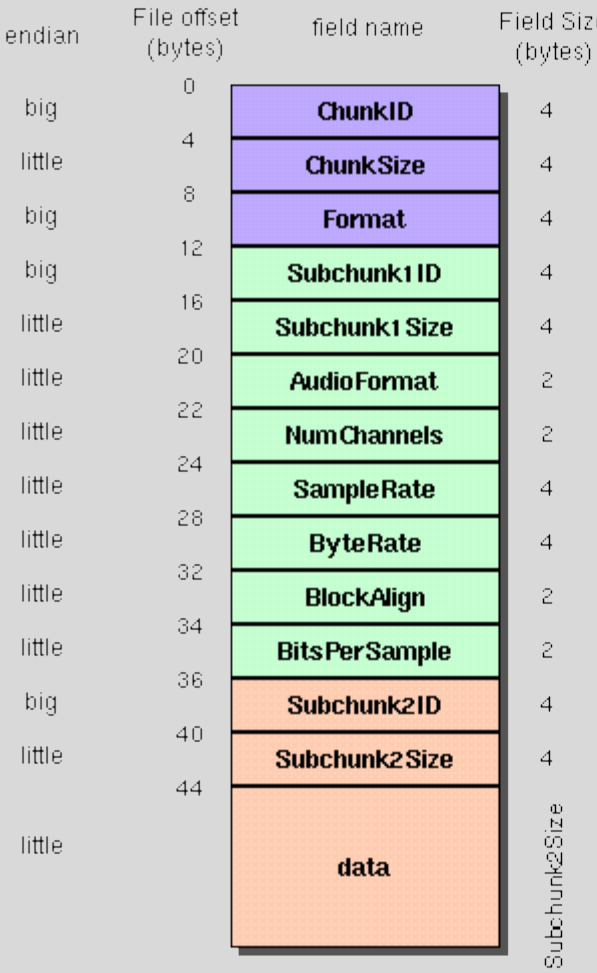


ESERCIZIO

Aprire il file WAV '16bit_stereo_8000Hz.wav' con un editor/viewer esadecimale (ad es. Textpad per Windows o 0xED per Mac), e ricercare le informazioni di header e chunk

ESERCIZIO

(16BIT_STEREO_8000HZ.WAV)



→ "RIFF" in formato ASCII, ossia 52 49 46 46 espresso in big endian

→ 64 01 00 00 = 164_{16} = 356 Byte = (364 – 8) Byte, c.v.d.

→ "WAVE" in formato ASCII, ossia 57 41 56 45 espresso in big endian

→ "fmt " in formato ASCII, ossia 66 6d 74 20 espresso in big endian

→ 10 00 00 00 = 16_{10}, valore tipico del PCM

→ 01 00 = 1_{16} = 1_{10}, ossia PCM (quantizzazione lineare)

→ 02 00 = 2_{16} = 2_{10}, ossia stereo

→ 40 1F 00 00 = $1F40_{16}$ = 8000_{10}, ossia 8000 Hz

→ 00 7D 00 00 = $7D00_{16}$ = 32000_{10} (è SampleRate · NumChannels · BitsPerSample / 8)

→ 04 00 = 4_{16} = 4_{10}, valore tipico del PCM (è NumChannels · BitsPerSample / 8)

→ 10 00 = 10_{16} = 16_{10}

→ "data" in formato ASCII, ossia 64 61 74 61 espresso in big endian

→ 40 01 00 00 = 140_{16} = 320_{10}, dimesione della parte rimanente

→ I veri e propri dati audio

LETTURA DI FILE BINARIO

Si ricorda come leggere il contenuto di un file e trasferirlo in strutture dati in memoria:

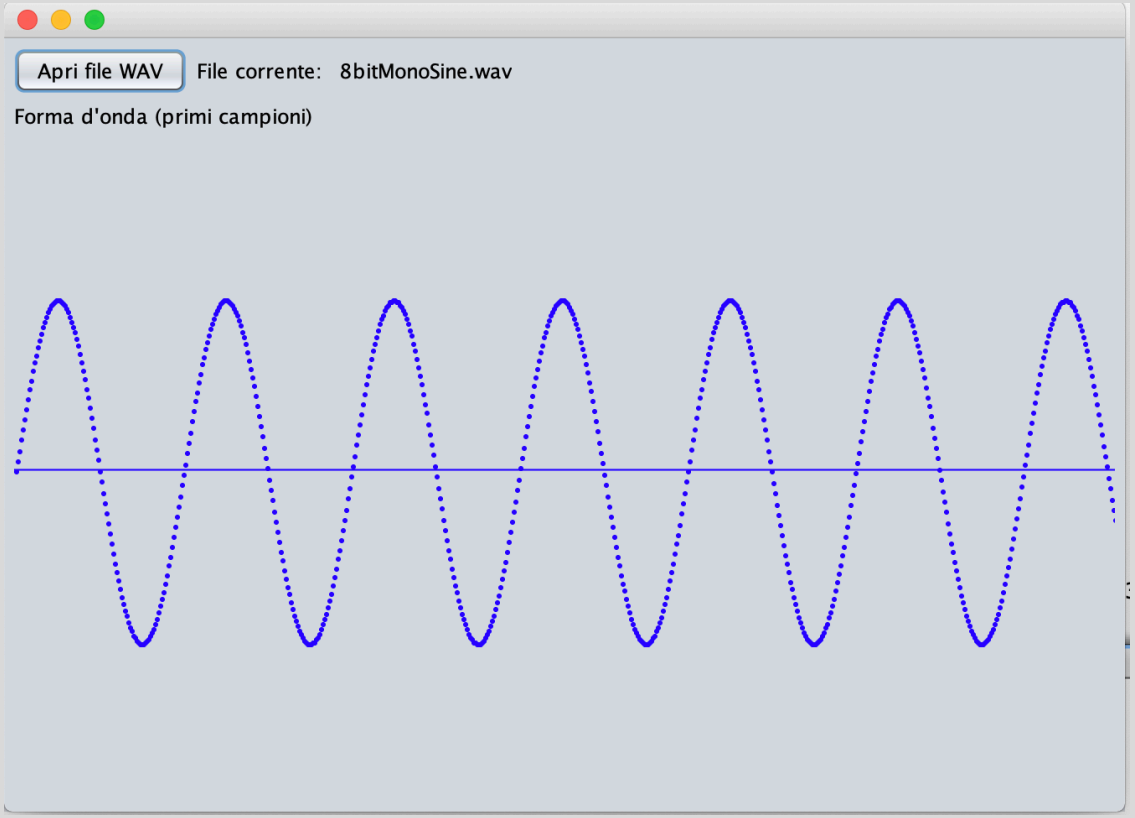
```
InputStream input = new FileInputStream(file);
int[] audioData = new int[(int) file.length()];
for (int i = 0; i < file.length(); i++)
    audioData[i] = input.read();
```

Si noti che `input.read()` legge un byte alla volta, ma il valore restituito è di tipo `int`

ESERCIZIO

(Wav8bitMonoReader.java)

- Scrivere un software con interfaccia grafica, che permetta di aprire un file WAV 8 bit Mono e disegni su un JPanel i valori dei primi campioni (partendo quindi dalla posizione 44 del file per escludere l'header canonico)



LETTURA E SCRITTURA DI FILE RAW

FORMATI RAW

- Nei formati RAW esiste solo il contenuto del chunk di dati, tipicamente non compresso
- Nel caso dell'audio, mancano dunque le informazioni di header che consentono di determinare frequenza di campionamento, numero di bit per campione, codifica little o big endian, numero di canali, ecc.
- Queste informazioni devono essere specificate dall'utente all'apertura del file da parte di un software in grado di supportare i formati RAW
- D'altro canto, la mancanza di qualsiasi forma di overhead implica un'estrema facilità nella lettura e scrittura dei campioni

LETTURA DI CAMPIONI DA FILE [1/2]

Principali passaggi:

1. Istanziare un oggetto di classe File con i contenuti audio
```
File fileToOpen = new File("onda_quadra_mono_16.raw");
```

2. Trasformare l'oggetto File in un oggetto InputStream
```
InputStream input = new FileInputStream(fileToOpen);
```

3. Trasformare l'oggetto InputStream in DataInputStream, che mette a disposizione metodi più efficaci per la lettura di una certa quantità di byte
```
DataInputStream data = new DataInputStream(input);
```

LETTURA DI CAMPIONI DA FILE [2/2]

Principali passaggi (continua):

4. Dichiarare e dimensionare un array di double della dimensione corretta per ospitare i valori dei campioni

```
int bufferSize = (int) (size * 8 / nbits);
double[] result = new double[bufferSize];
```

   ove size è la dimensione del file originario

5. Leggere la quantità opportuna di bit (e con il segno espresso correttamente) dallo stream al fine di valorizzare l'array

Limite: questa tecnica si applica a valori a 8 bit o espressi in big endian

IL PROBLEMA DELLA ENDIANNESS

- La lettura dello stream senza riordinare i byte (in caso di codifiche little endian) introduce evidenti errori nel caso in cui i valori dei campioni si esprimano su più di 8 bit

- Soluzione: leggere lo stream byte per byte imponendo un corretto riordino dei byte, a seconda delle caratteristiche del campionamento. Allo scopo è possibile:
  1. utilizzare la classe ByteOrder (che contiene un'enumerazione per differenziare little e big endian)
  2. Sfruttare il metodo statico wrap() della classe ByteBuffer + metodo get()

```
byte[] bytes = {…};                          // caricamento del file byte per byte
short[] samples = new short[bytes.length / 2]; // 2 = bitDepth / 8
ByteBuffer.wrap(bytes).order(ByteOrder.LITTLE_ENDIAN)
                                    .asShortBuffer().get(samples);
```

SCRITTURA DI CAMPIONI SU FILE

- Nel caso di stream di campioni da 8 bit, o da codificare in big endian, è sufficiente sfruttare la classe FileOutputStream

```
FileOutputStream fos = new FileOutputStream(file);
fos.write(samples);
fos.close();                    // invocare close() richiama implicitamente flush()
```

- In analogia con quanto visto per la lettura di campioni, il principale problema della scrittura consiste nel riordinare correttamente i byte nel caso di approccio little endian

```
short[] samples = {…};
byte[] bytes = new byte[samples.length * 2];
ByteBuffer.wrap(bytes).order(ByteOrder.LITTLE_ENDIAN)
                                        .asShortBuffer().put(samples);
```

SCRITTURA DI CAMPIONI SU FILE

```
short[] samples = {…};
byte[] bytes = new byte[samples.length * 2];
ByteBuffer.wrap(bytes).order(ByteOrder.LITTLE_ENDIAN)
                                .asShortBuffer().put(samples);
```

Una volta ottenuto l'array di byte ordinato con l'endianness corretta, questo va trasferito su disco

Metodo più diretto: classe FileOutputStream

```
FileOutputStream fos = new FileOutputStream(mioFile);
fos.write(bytes);
fos.close();
```

Oppure: classe BufferedOutputStream

```
BufferedOutputStream bos = new BufferedOutputStream(new
                                        FileOutputStream(mioFile));
bos.write(bytes);
bos.flush();
bos.close();
```

CONVERSIONE DI VALORI DA SHORT A BYTE

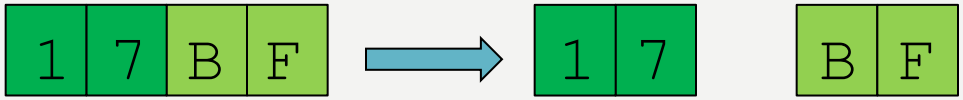
- La struttura dati interna prescelta per la rappresentazione dei campioni dovrà essere dimensionata con il tipo più opportuno, a seconda della bit depth (il numero di bit selezionato per la quantizzazione)

- Ad esempio, una struttura dati adeguata per un file in qualità CD-audio (44100Hz, 16 bit per campione, 2 canali) potrebbe essere:
  - un doppio array di short (uno per il canale sinistro, uno per il canale destro)
  - un ArrayList di coppie di short
  - …

- Problema: trasformare tale struttura dati in una sequenza correttamente ordinata di byte (problema triviale se si lavora in mono a 8 bit). Il data packing è già stato illustrato, ma come si convertono gli short in byte little endian?

DA SHORT A BYTE: APPROCCIO MATEMATICO

- Obiettivo: trasformare uno short, ossia un valore a 16 bit, in due byte, ossia due valori indipendenti a 8 bit, corrispondenti al byte di ordine alto e di ordine basso



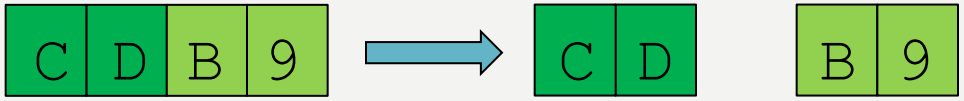
- Soluzione: basarsi su divisione intera e resto della divisione intera per opportune potenze di due

- Esempio:
  $17BF_{16} = 6079_{10}$
  $6079 / 2^8 = 6079 / 256 = \text{floor}(23{,}7460\ldots) = 23_{10} = 17_{16}$
  $6079 \% 2^8 = 191_{10} = BF_{16}$

- Si può generalizzare alla conversione da int: i 4 byte si ottengono dividendo il valore per 2^{24}, 2^{16}, 2^8 e considerando infine il resto

DA SHORT A BYTE: APPROCCIO MATEMATICO

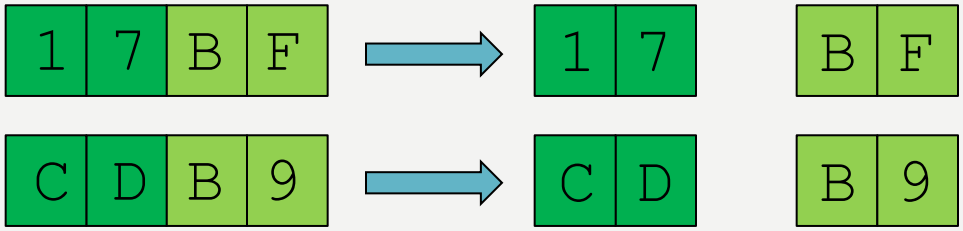
- Problema: questo approccio si basa su operazioni aritmetiche, per cui intrinsecamente non può prescindere da come viene interpretato il valore aritmetico
  Poiché viene adottata la notazione in complemento a 2, i valori superiori a $2^{n-1}-1$ vengono letti come valori negativi, e questo inficia l'approccio



- Esempio:
  $CDB9_{16} = 52665_{10}$ ma viene interpretato come -12871_{10}
  $-12871 / 2^8 = -50_{10} = FFFFFFCE_{16} = CE_{16} \rightarrow$ Errato!!!
  $-12871 \% 2^8 = -71_{10} = FFFFFFB9_{16} = B9_{16}$

DA SHORT A BYTE: APPROCCIO INFORMATICO

- Obiettivo: trasformare uno short, ossia un valore a 16 bit, in due byte, ossia due valori indipendenti a 8 bit, corrispondenti al byte di ordine alto e di ordine basso

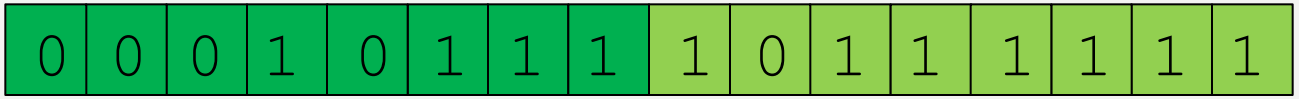


- Soluzione: operare opportuni bitshift (in Java operatore >>) e selezionare quelli significativi tramite una maschera di bit (in Java operatore &, ossia «and bit a bit»)

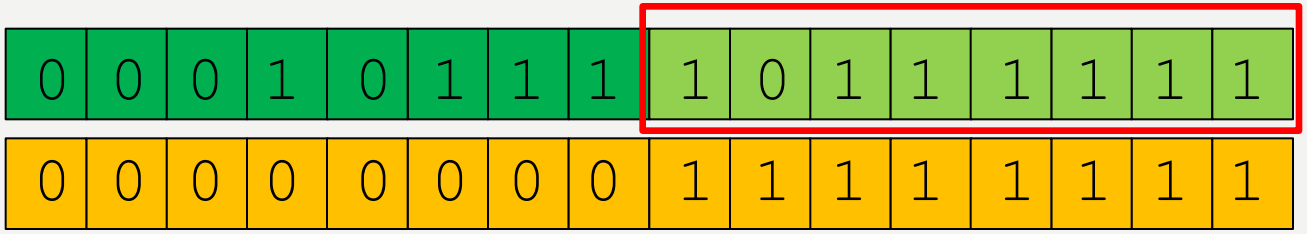
- In questo caso si deve prescindere dall'interpretazione numerica del valore espresso su 16 bit. È logico lavorare direttamente sui bit

DA SHORT A BYTE (ESEMPIO)

Esempio 1: $17BF_{16} = 0001011110111111_2$

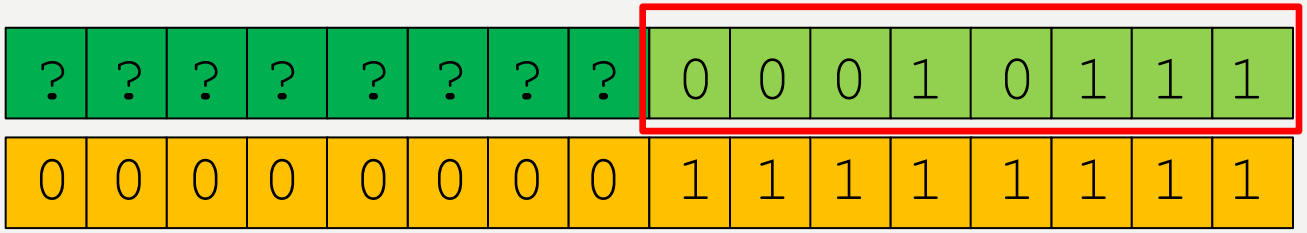


Byte di ordine inferiore: s & 0xff



BF_{16}

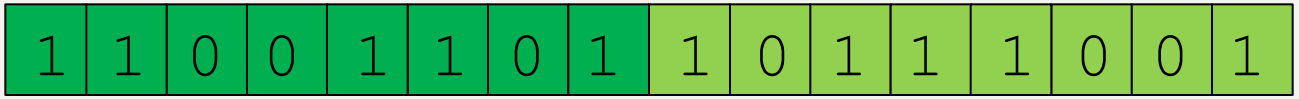
Byte di ordine superiore: (s >> 8) & 0xff



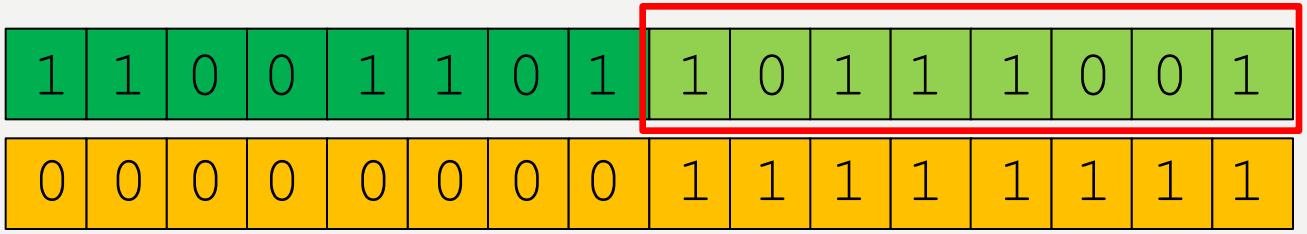
17_{16}

DA SHORT A BYTE (ESEMPIO)

Esempio 1: $CDB9_{16} = 1100110110111001_2$

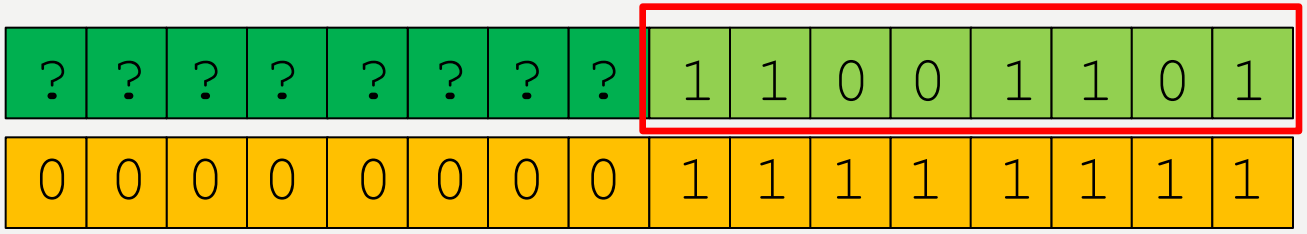


Byte di ordine inferiore: s & 0xff



$B9_{16}$

Byte di ordine superiore: (s >> 8) & 0xff



CD_{16}

ESEMPI

- Apertura di file RAW a 8 bit e a 16 bit contenenti onde quadre e sinusoidali, con visualizzazione dei valori numerici assunti dai singoli campioni

Per comprendere quanto avviene nel software, si suggerisce di aprire i 3 file in Audacity (Importa dati RAW) e impostare i parametri del formato RAW con la profondità di bit corretta, con particolare attenzione all'endianness

Si noterà che:

- La forma d'onda sinusoidale del file a 8 bit chiaramente non cambia
- La forma d'onda quadra a 16 bit sembra non cambiare, ma si tratta di un caso: per via dell'ampiezza impostata, i campioni si attestano intorno a $26470_{10} = 67\ 66_{16}$ e quindi scambiare l'ordine dei 2 byte (big endian vs. little endian) non cambia molto
- La forma d'onda sinusoidale del file a 16 bit mostra evidenti differenze se aperta in codifica big endian o little endian

- Software per la scrittura di stream a 8 e 16 bit sul disco, considerando l'endianness

ESERCIZIO

(InvertEndianness.java)

- Si scriva un software che agisce su file audio in formato PCM RAW al fine di invertire l'endianness (ossia trasforma un big endian in little endian e viceversa)
- Il risultato andrà salvato su disco con un nome differente
- All'utente sarà richiesto di specificare la profondità di bit (ossia quanti bit occupa un campione), il cui valore sarà multiplo di 8
- Si utilizzi un editor di forma d'onda quale Audacity per verificare che il risultato dopo l'elaborazione dei byte sia coerente con il documento iniziale

GENERAZIONE DI FORME D'ONDA

DA RAW A WAV

- Rispetto ai formati audio RAW, nei formati WAV è presente un header
- Si ricorda che il formato dell'header comporta l'inserimento di 44 byte all'inizio del file, cui vengono poi aggiunti in coda i dati audio veri e propri, ossia il contenuto dell'intero file RAW corrispondente
- Nel seguito delle lezioni, si farà principalmente riferimento a un file audio WAV con caratteristiche Audio CD, ossia campionato a 44100 Hz e con 16 bit a campione. Il numero di canali sarà 1 o 2 a seconda dei casi

ESEMPIO

(WavHeader.java)

- Creazione di un header con caratteristiche specifiche fissate dall'utente per l'aggiunta ad un file RAW, convertendo in WAV

- Il risultato opposto, ossia rendere RAW un file WAV, si ottiene semplicemente leggendo i byte contenuti nel file WAV a partire dall'indirizzo 44 in avanti

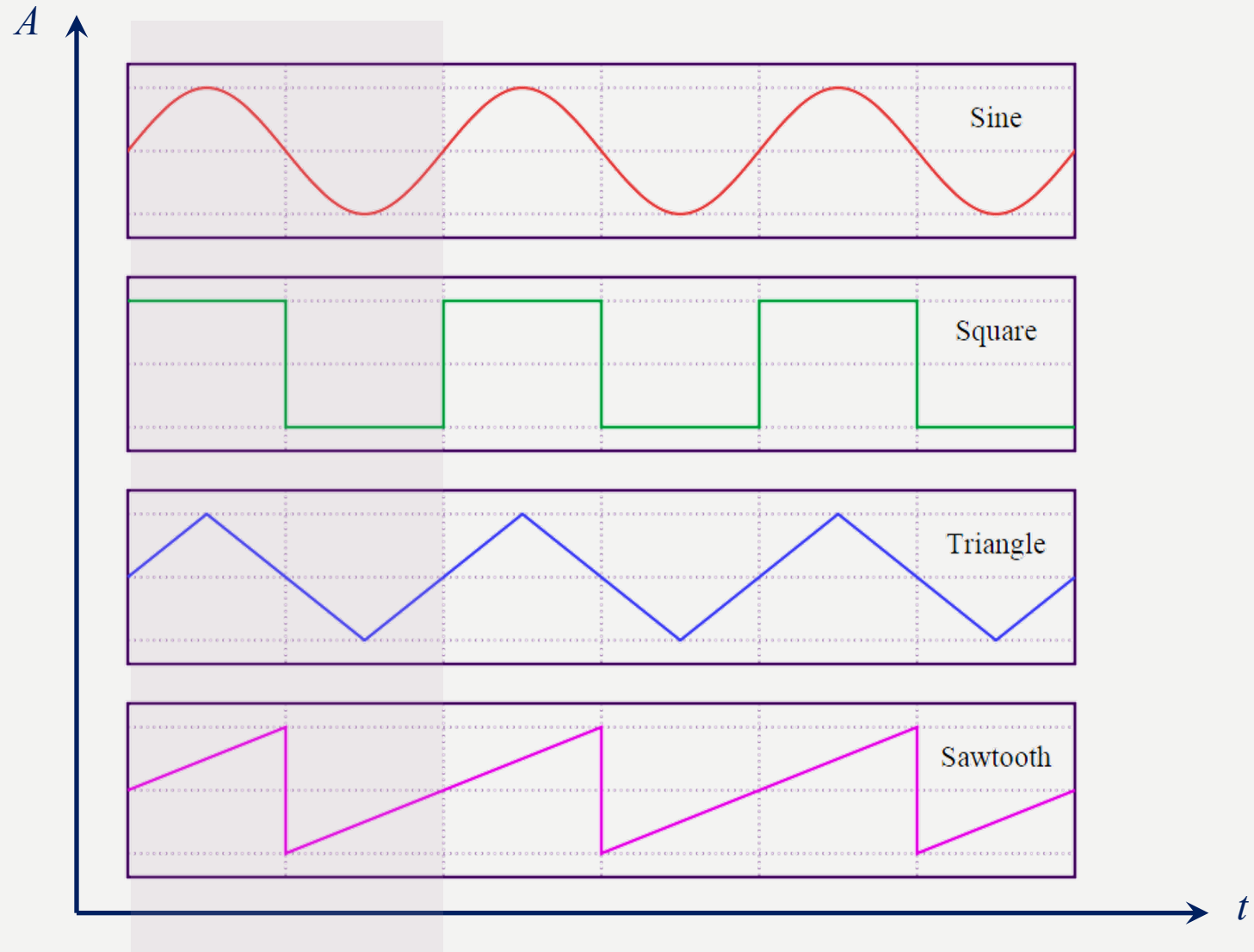
SINTESI DIGITALE DEL SUONO

- Una volta stabilita una struttura dati opportuna, è possibile generare forme d'onda periodiche agendo per via numerica

- Questa operazione è particolarmente semplice per forme d'onda che rispondono a semplici equazioni, quali la sinusoide, l'onda quadra, il dente di sega, ecc.

- Per ciascuna di queste, fissata a priori la frequenza di campionamento e la forma d'onda, è necessario specificare frequenza e ampiezza

ALCUNE FORME D'ONDA NOTEVOLI



ESEMPI

(CreateWav1.java)

Creazione di una classe per svolgere le seguenti operazioni sulla forma d'onda:

- Rappresentazione dei campioni per i canali sinistro e destro
- Aggiunta di un header con caratteristiche specifiche fissate dall'utente ad un file RAW
- Salvataggio
- Generazione di rumore bianco, onda sinusoidale e dente di sega

ESERCIZIO

(CreateWav2.java)

- Partendo dall'esempio precedente, arricchire la libreria di funzioni generatrici aggiungendo l'onda quadra e triangolare

PROCESSORI DI DINAMICA

ELABORAZIONE DELLA DINAMICA DEL SEGNALE AUDIO

- Nel momento in cui il segnale audio (discretizzato) è caricato nelle strutture dati in memoria, è possibile – e relativamente semplice - agire algoritmicamente su una parte o su tutti i campioni

- Il valore numerico assunto dal campione ne rappresenta il valore di ampiezza, quantizzato e (plausibilmente) espresso in valori lineari, normalizzato o meno
  - Ad esempio, se la quantizzazione è a 16 bit, i valori ricadono nell'intervallo [-32768..32767]

OPERAZIONI COMUNI SULLA DINAMICA

- Regolazioni di volume e normalizzazione
  - La normalizzazione audio è il processo di aumento o riduzione dell'ampiezza di un segnale audio. Di solito il processo consiste nel rendere massimo il livello di un segnale audio senza introdurne distorsione
- Compressione/espansione della gamma dinamica in base a un valore di soglia (threshold)
  - L'utilizzo di una funzione distorcente per un segnale audio permette di effettuare la cosiddetta sintesi per distorsione non lineare o waveshaping
- Clipping e foldback

FUNZIONE DISTORCENTE

- Nella sua forma più controllabile, il waveshaping si basa sulla lettura di una tabella che contiene una funzione apposita, detta funzione distorcente

- Si tratta di calcolare e/o rappresentare una funzione

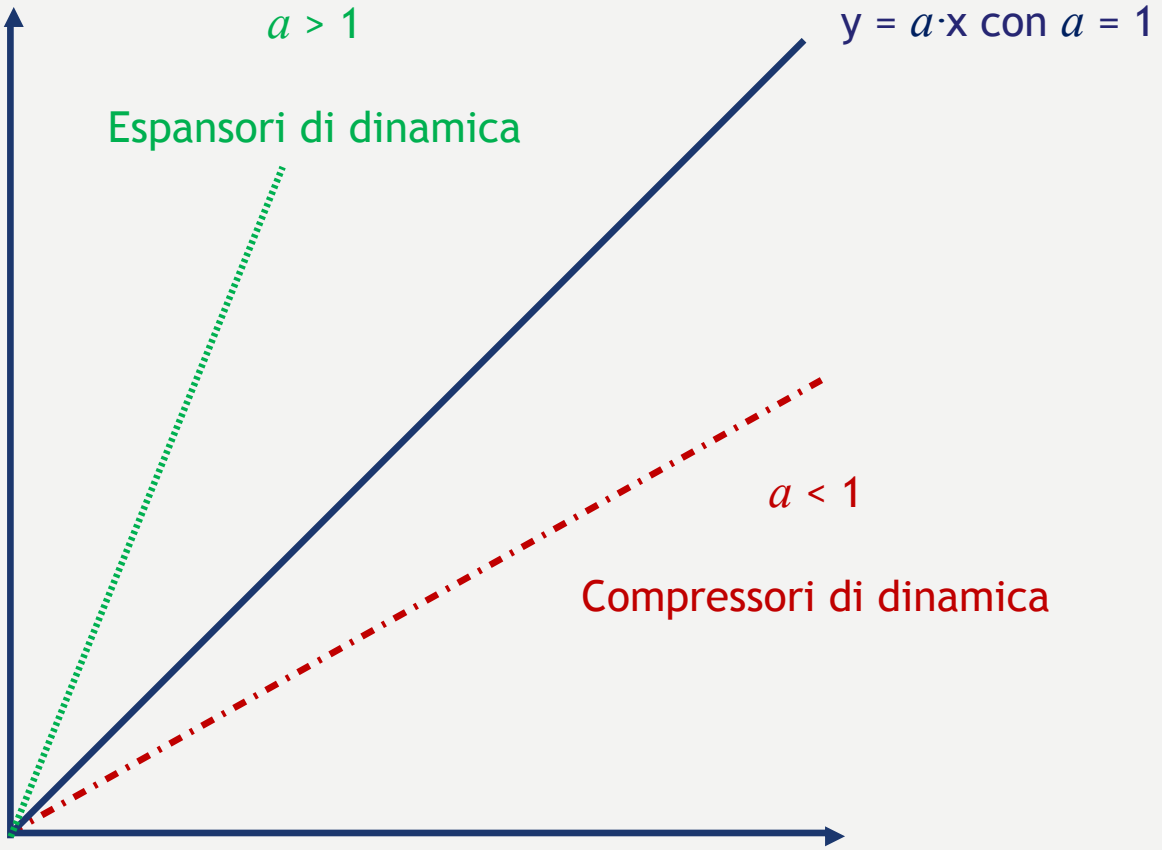
  $y = f(x)$, ove x in questo caso è il valore di ampiezza istantaneo del segnale audio entrante, mentre y rappresenta il nuovo valore di ampiezza da conferire al segnale audio in corrispondenza di tale ingresso per eseguire il waveshaping

CASISTICA

- Se la funzione è una retta non ha luogo distorsione:
  - se la retta è inclinata a 45° e passa per l'origine, cioè y = f(x) = x allora il valore di uscita y è pari all'ingresso x
  - se la retta ha inclinazione diversa e passa dall'origine, cioè y = f(x) = a·x allora l'andamento dell'uscita è uguale a quello dell'ingresso, salvo un fattore correttivo (che può essere maggiore o minore di 1) > vedi compressori ed espansori di dinamica
  - se la retta non passa per l'origine, ossia y = f(x) = a·x + b allora valgono i discorsi precedenti, ma con l'aggiunta di un offset

- Se non si tratta di una retta, si ha distorsione: questo discorso applicato a un segnale audio implica l'introduzione di nuove frequenze

ESEMPIO GRAFICO DI NON DISTORSIONE



ESPANSORI DI DINAMICA

- Un espansore è un dispositivo che, per una certa ampiezza efficace del segnale in ingresso, fa variare di una quantità maggiore il segnale in uscita ($y = a \cdot x$ con $a > 1$)

- Un espansore può servire per ampliare la gamma dinamica di un segnale che risulta troppo schiacciato e per restituirgli aggressività

COMPRESSORI DI DINAMICA

- Un compressore è un dispositivo che, per una certa ampiezza efficace del segnale in ingresso, fa variare di una quantità minore il segnale in uscita ($y = a \cdot x$ con $a < 1$)

- Un compressore può servire per limitare la gamma dinamica di un segnale, ad esempio per evitare che i suoi picchi mandino in distorsione il sistema

- Esistono anche effetti musicali basati sui compressori, come ad esempio la compressione molto spinta in uso nel rock che neutralizza l'attacco delle note, eliminando l'effetto di corda pizzicata sulle chitarre

COMPRESSIONE E LIMITAZIONE

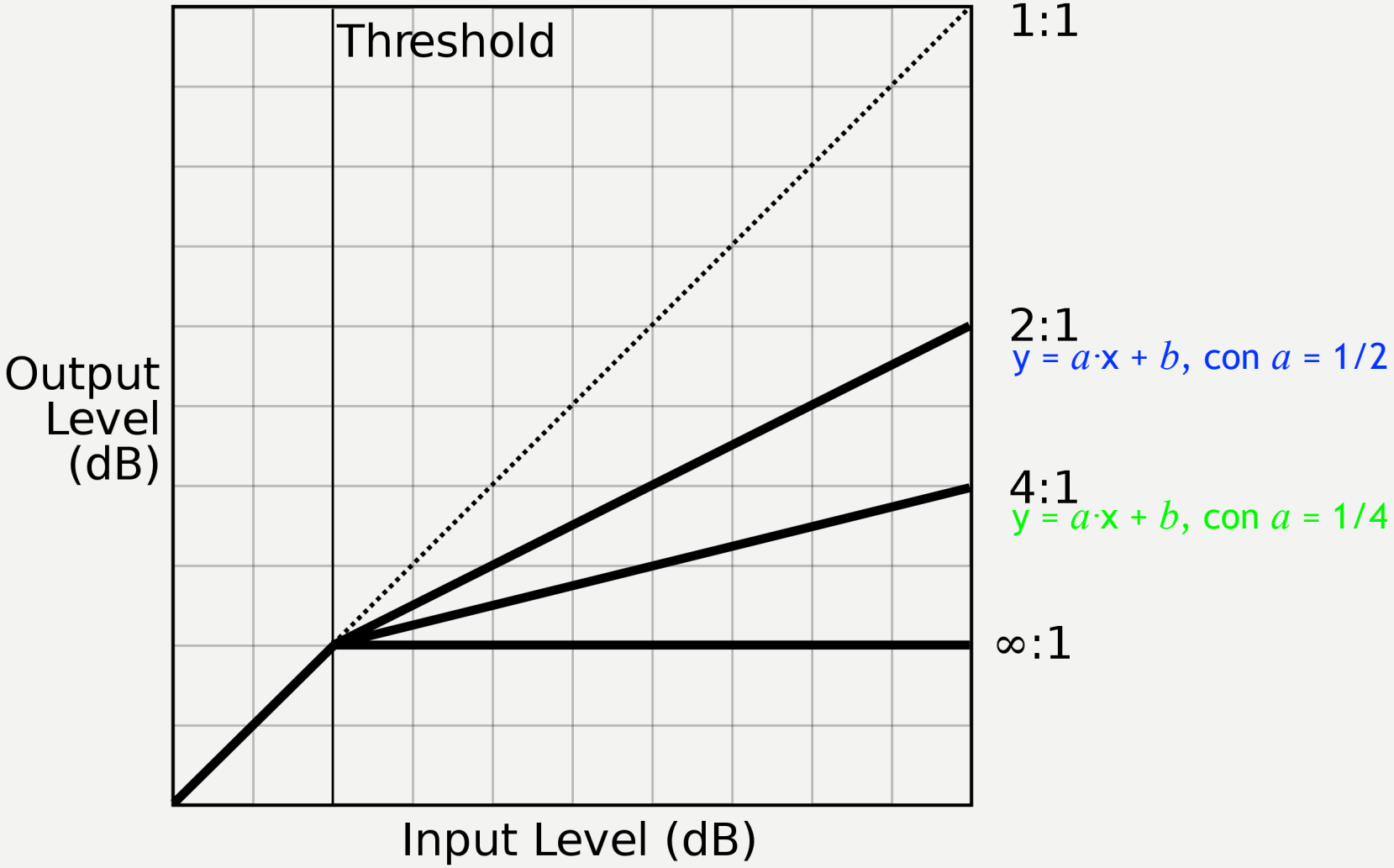
- Nell'ambito delle infinite funzioni non lineari che possono definire il comportamento di un compressore, vi sono alcune definizioni standard

- Si definisce soglia di compressione il livello in cui un guadagno unitario (y = x) si trasforma in una retta con pendenza minore, e il rapporto di compressione è per definizione $(1/a) : 1$

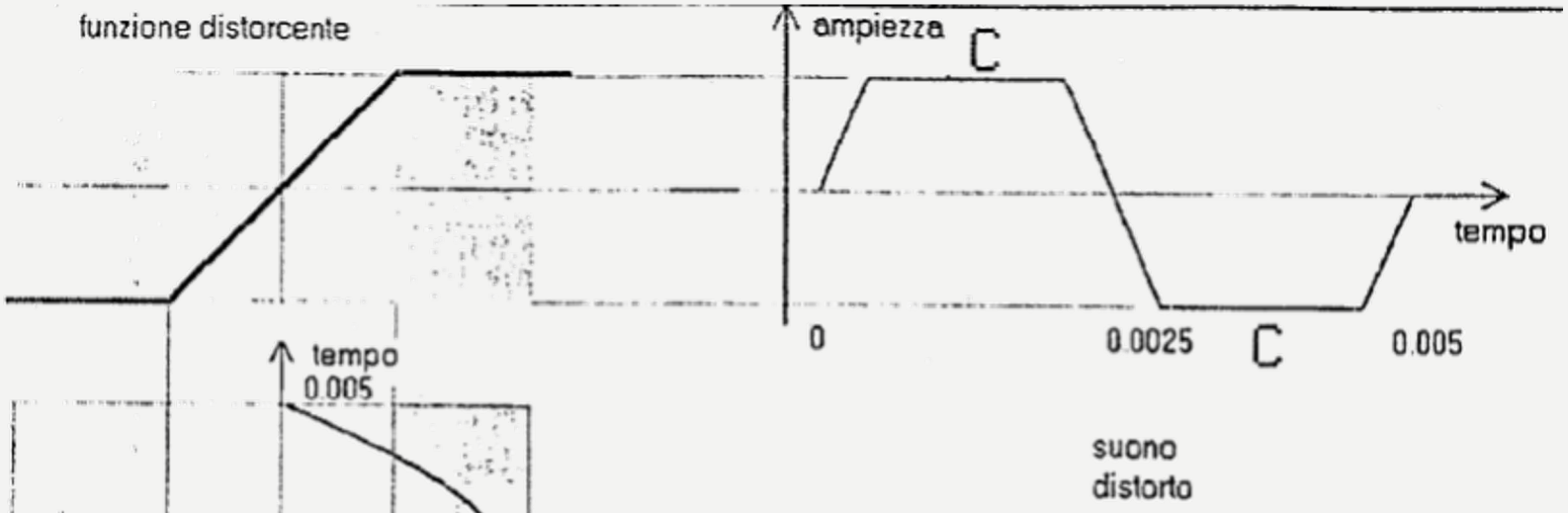
- Si definisce soglia di limitazione il livello oltre il quale la retta assume pendenza nulla

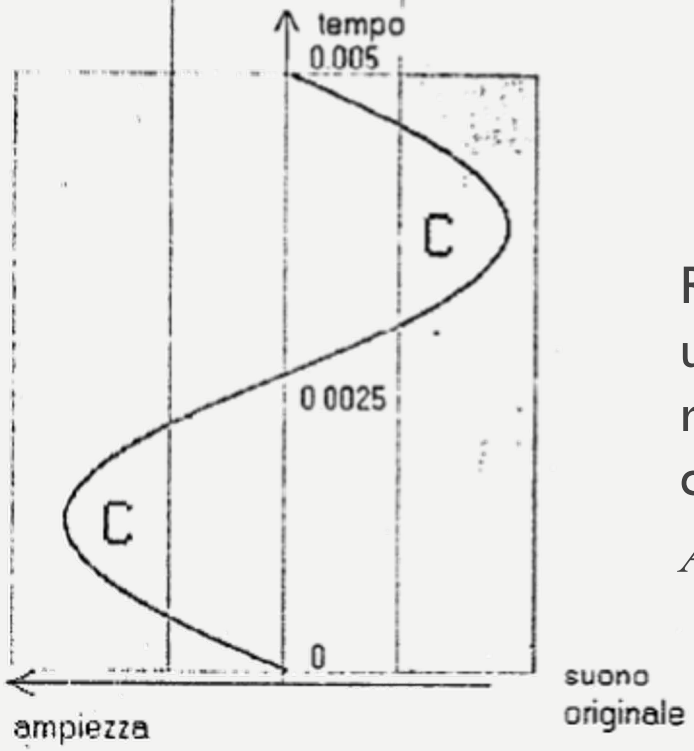
- Spesso in seguito alla compressione si usa un fattore moltiplicativo correttivo per compensare la diminuzione globale di volume

ESEMPIO DI DISTORSIONE NON LINEARE



ALTRO ESEMPIO DI DISTORSIONE NON LINEARE





Risultato: la sinusoide in ingresso diventa in uscita una sinusoide tagliata al raggiungimento delle ampiezze $+A$ e $-A$, con conseguente modifica dello spettro

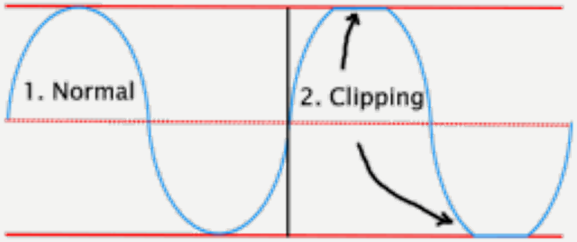
A rappresenta il livello di clipping

EFFETTI DI DISSOLVENZA

- Il volume di uno o più canali può anche essere fatto variare in funzione del tempo

- Una delle applicazioni tipiche è l'effetto di fade-in (dissolvenza in ingresso) e fade-out (dissolvenza in uscita)

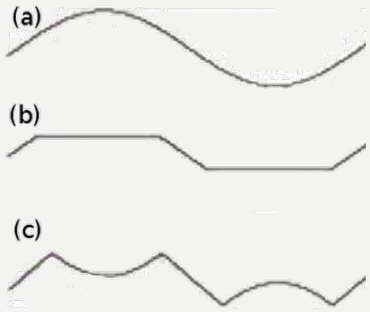
CLIPPING



- Quando il segnale viene spinto alla massima ampiezza (ossia satura i bit dedicati alla quantizzazione) e non può essere amplificato ulteriormente, ha luogo un taglio o clip, e il segnale è detto essere in clipping

- Il segnale extra può essere semplicemente tagliato, con una risultante tipica assimilabile ad un'onda quadra
  - Ad esempio alcuni chitarristi con chitarra elettrica e amplificatore intenzionalmente sovraccaricano l'amplificatore per portarlo al clipping, ottenendo un suono volutamente distorto (overdrive)

FOLDBACK



- Un'altra forma di distorsione che attua un taglio della forma d'onda ad un livello prestabilito di ampiezza è il foldback

- In questo caso, la forma d'onda «in eccesso» non viene eliminata, ma viene riflessa specularmente nel semipiano opposto, ottenendo un effetto di ripiegamento su se stessa

- In figura:
  - a) segnale originale
  - b) clipping
  - c) foldback

ESEMPIO

(WavUtils1.java)

- Creazione di una classe con metodi per processare la dinamica:
  - Regolazione del volume (clipping)
  - Distorsione non lineare tramite overdrive
  - Normalizzazione

ESERCIZIO

(WavUtils2.java)

- Modificare il codice precedente aggiungendo un metodo per attuare il foldback

SINTESI WAVETABLE

SINTESI MEDIANTE FORMA D'ONDA FISSA

- La caratteristica di molti suoni musicali è di essere periodici o quasi-periodici

- Il più semplice metodo di sintesi consiste nel produrre un segnale periodico mediante la ripetizione continua di una certa forma d'onda

- Questo metodo viene chiamato sintesi con forma d'onda fissa, e il modulo che la realizza si chiama oscillatore digitale

WAVETABLE E TABLE LOOKUP

- I valori di ampiezza che corrispondono alla discretizzazione opportuna della forma d'onda:
  - possono essere calcolati in modo puntuale, campione per campione, sulla base di funzioni matematiche generatrici (come già illustrato in precedenza)
  - possono essere letti da una tabella (wavetable), in cui i valori di ampiezza pre-calcolati vengono tabulati per essere poi letti ciclicamente (metodo del table lookup)
  - possono richiedere interventi più complessi, come nel caso della sintesi a wavetable multiple

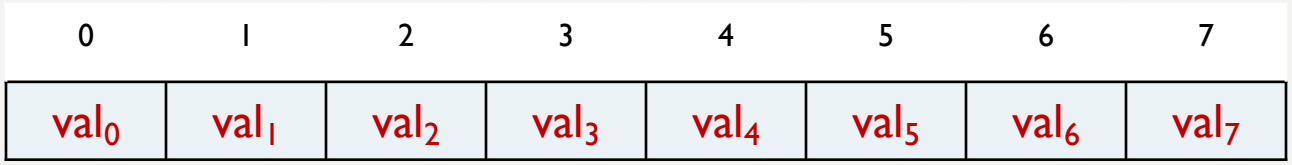
PERIODO E CICLO

- Il parametro fondamentale di una forma d'onda periodica è il periodo, ossia l'intervallo di tempo in cui si completa un ciclo

- È possibile fissare il numero di punti della tabella, ossia il numero di celle al cui interno verranno scritti i valori provenienti dal campionamento della forma d'onda (in questo caso, di un singolo ciclo)

- Tanto maggiore sarà il numero di punti, tanto più raffinata sarà la discretizzazione

LETTURA DI CAMPIONI DA UNA TABELLA

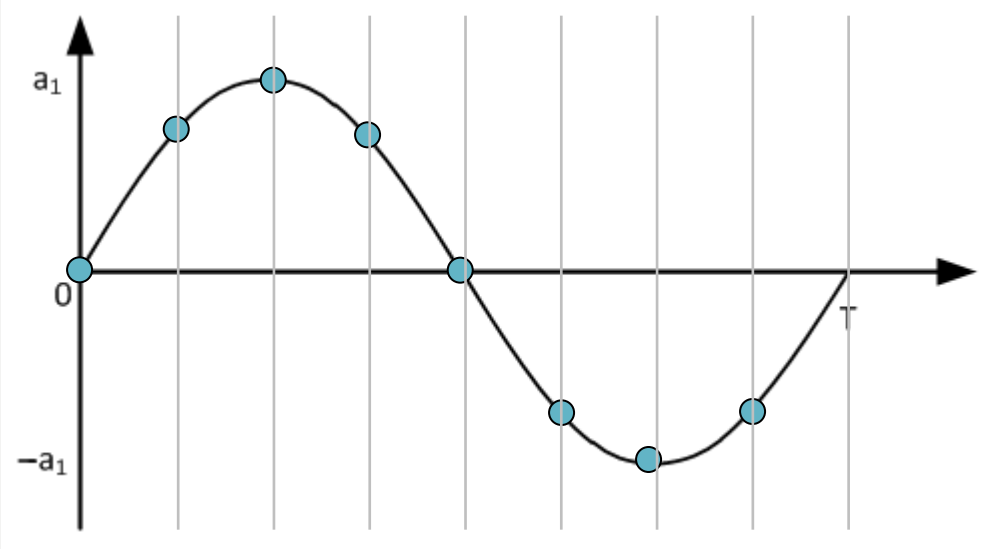


- Nella conversione analogico-digitale di una forma d'onda, tipicamente si fissa un'opportuna frequenza di campionamento e si determina così un certo numero di valori nell'unità di tempo

- Per la sintesi con forma d'onda fissa, è già disponibile un numero di punti prefissato. Il problema diventa a che velocità l'oscillatore digitale deve leggere la tabella per produrre una data frequenza

ESEMPIO

- Sia data una tabella contenente *n* valori per discretizzare un ciclo.
  - In figura, il ciclo è un'oscillazione sinusoidale pura e *n* = 8 (valore molto basso per ottenere una buona ricostruzione)



ESEMPIO

- Ipotesi di lavoro:
  - file audio con frequenza di campionamento 16 KHz (16000 punti al secondo)
  - all'oscillatore digitale si richiede una frequenza di 2 KHz

- L'oscillatore deve «riempire» 16000 punti al secondo con 2000 oscillazioni complete al secondo
  Ogni oscillazione richiede quindi di compilare
  16000 / 2000 = 8 punti, che sono esattamente quelli presenti in tabella

- Soluzione: l'oscillatore legge dalla tabella valore per valore

ESEMPIO

- Ipotesi di lavoro:
  - file audio con frequenza di campionamento 16 KHz (16000 punti al secondo)
  - l'oscillatore digitale deve emettere una frequenza di 4 KHz

- In tabella è presente il doppio dei valori richiesti, perché la durata del ciclo da produrre è 16 / 4 = 4 punti

- Soluzione: l'oscillatore legge dalla tabella solo un valore su due

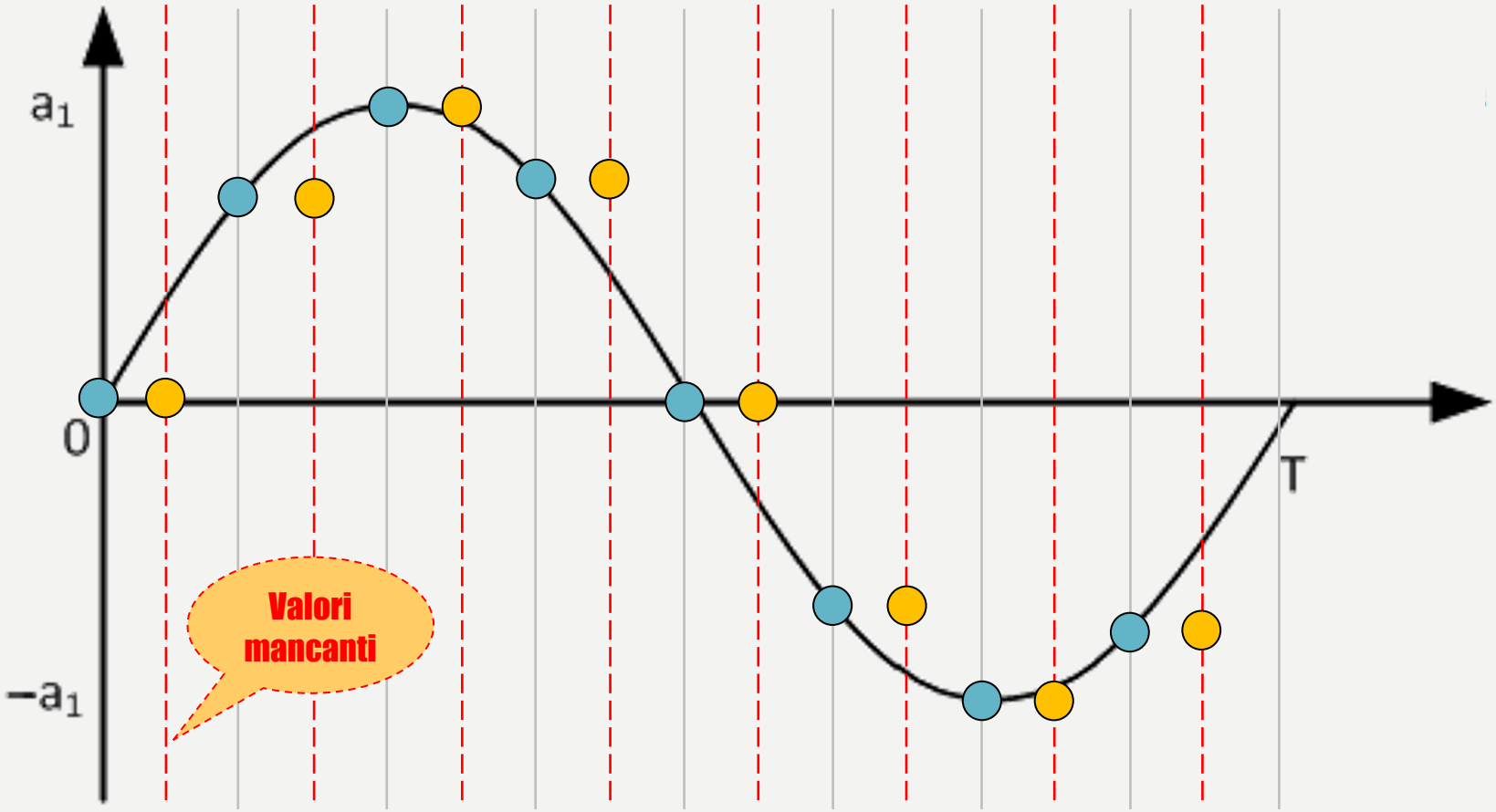
ESEMPIO

- Ipotesi di lavoro:
  - file audio con frequenza di campionamento 16 KHz (16000 punti al secondo)
  - l'oscillatore digitale deve emettere una frequenza di 1 KHz

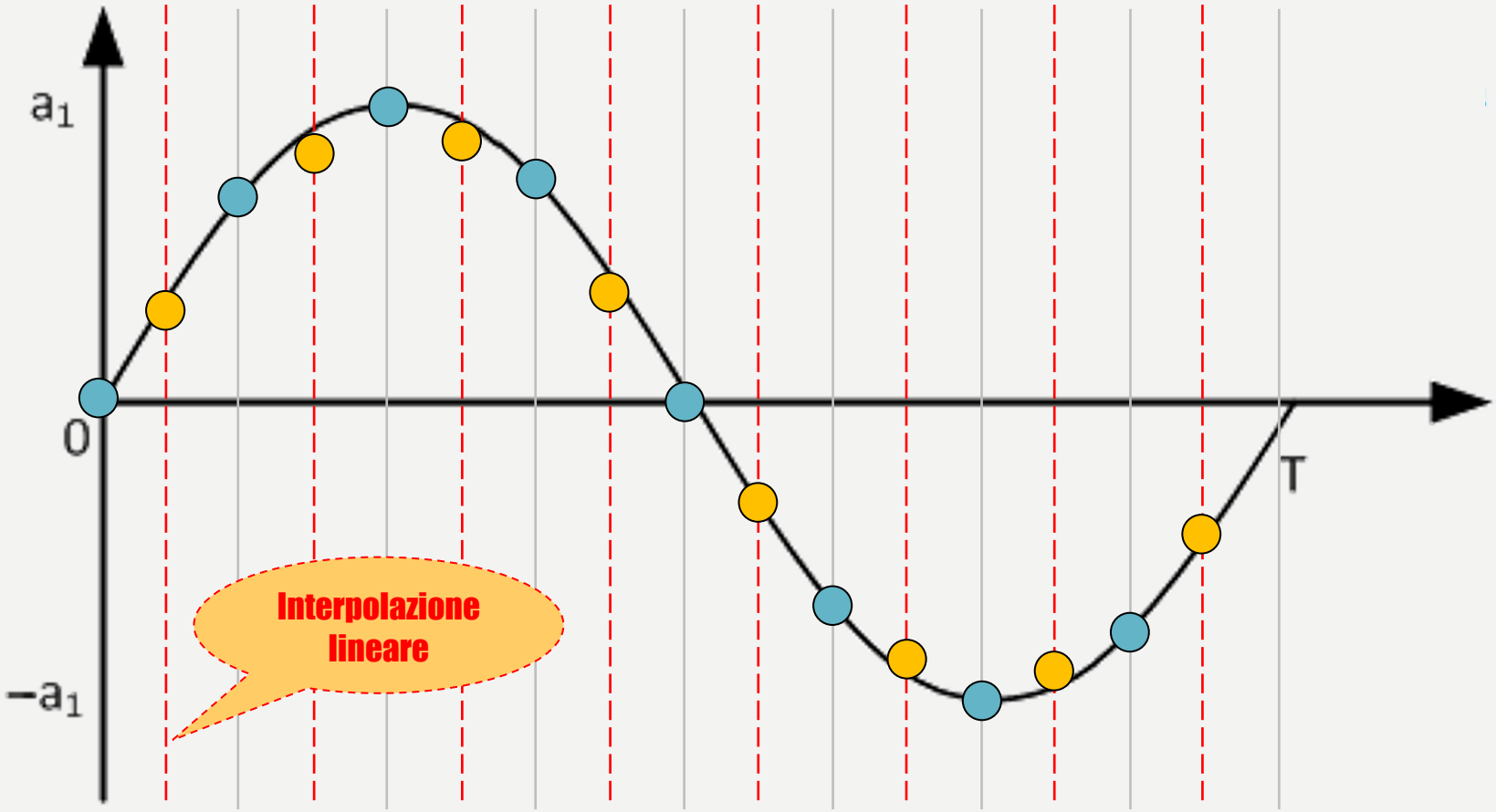
- In tabella è presente la metà dei valori richiesti, perché la durata del ciclo nello stream audio da produrre è 16 / 1 = 16 punti

- Soluzione: l'oscillatore deve leggere ed emettere anche valori non presenti in tabella. Come fare?

LA SOLUZIONE SENZA INTERPOLAZIONE



LA SOLUZIONE CON INTERPOLAZIONE



INTERPOLAZIONE: VANTAGGI E SVANTAGGI

- Le due tecniche di calcolo dei valori mancanti, con e senza interpolazione, funzionano anche quando il numero di punti da calcolare (ossia non presenti in tabella) non è un multiplo intero del numero di punti disponibili

- L'interpolazione lineare provoca un errore di quantizzazione minore (o al più uguale), quindi fornisce risultati migliori ma con tempi di calcolo maggiori

APPLICABILITÀ DELLA SINTESI WAVETABLE

- Abbiamo illustrato l'approccio per suoni periodici, ossia scomponibili in cicli elementari della durata di un periodo T che vengono ripetuti *n* volte. Nello specifico, abbiamo portato l'esempio del ciclo di sinusoide

- La sintesi wavetable è applicabile a qualsiasi forma d'onda ciclica (onda quadra, dente di sega, o cicli dalla forma maggiormente complessa)

- Una wavetable può inoltre contenere la discretizzazione di una forma d'onda non periodica

WAVETABLE PER SUONI CAMPIONATI

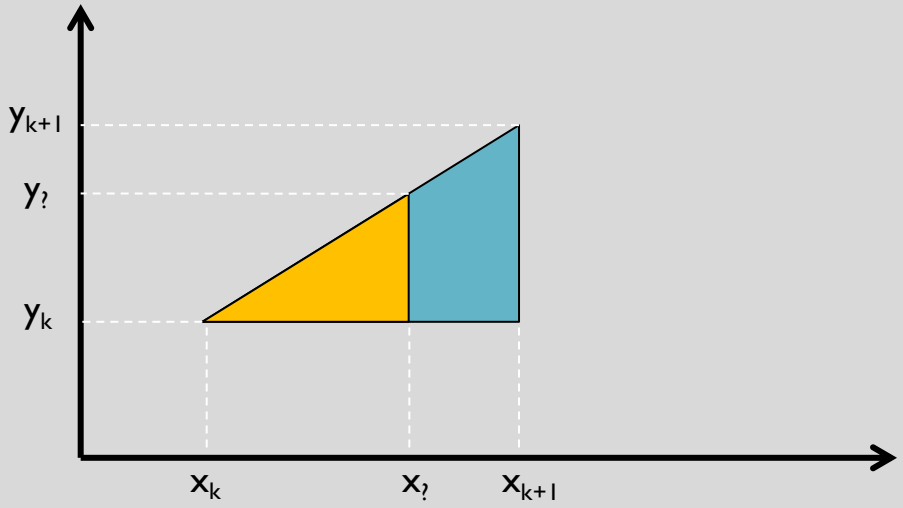
- Essendo una tecnica onerosa in termini di occupazione in memoria, le wavetable vengono impiegate solitamente per suoni singoli e di breve durata (ad es. una nota di pianoforte)

- Se il suono tabulato non è ciclico, la sua discretizzazione è legata a una data frequenza (ad es. la discretizzazione in 4096 campioni di un La a 440Hz)

ESEMPIO

(TestWavetableAlternatives.java)

- Confronto tra le diverse tecniche di generazione di una forma d'onda periodica, tra cui la table lookup con e senza interpolazione

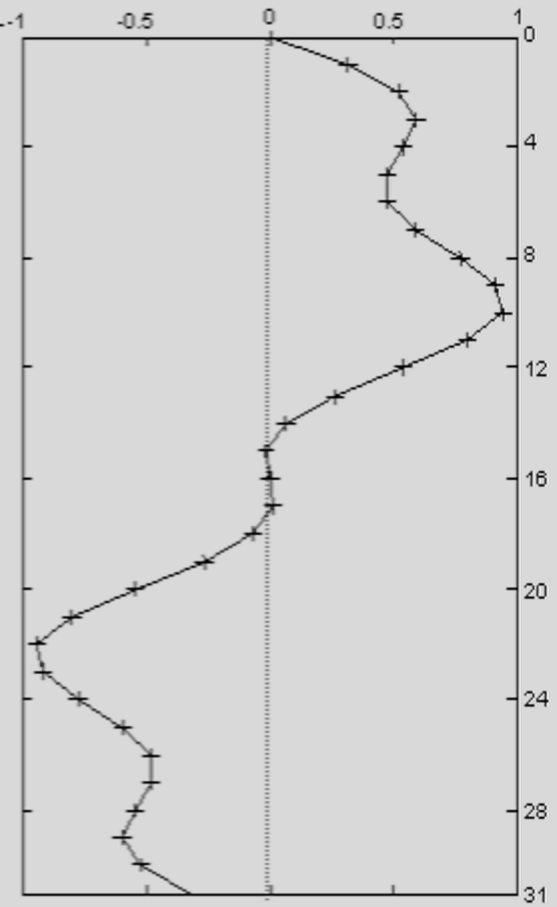


ESERCIZIO

(CustomWavetable.java)

- Nel documento di testo chiamato wavetable.txt si elencano, uno per riga, i valori della colonna "Sample" riportati a fianco, corrispondenti a una wavetable a 32 campioni
- Si realizzi un software in grado di caricare i contenuti del file e di effettuare sintesi wavetable (con interpolazione lineare) rispetto ad ampiezza e frequenza data
- Salvare il risultato su file in formato RAW 22KHz 8bit mono

Index	Sample
0	0.000
1	0.313
2	0.525
3	0.591
4	0.544
5	0.476
6	0.480
7	0.591
8	0.769
9	0.918
10	0.941
11	0.803
12	0.544
13	0.264
14	0.064
15	-0.013
16	0.000
17	0.013
18	-0.064
19	-0.264
20	-0.544
21	-0.803
22	-0.941
23	-0.918
24	-0.769
25	-0.591
26	-0.480
27	-0.476
28	-0.544
29	-0.591
30	-0.525
31	-0.313



MODULAZIONE D'AMPIEZZA E FREQUENZA

MODULAZIONE

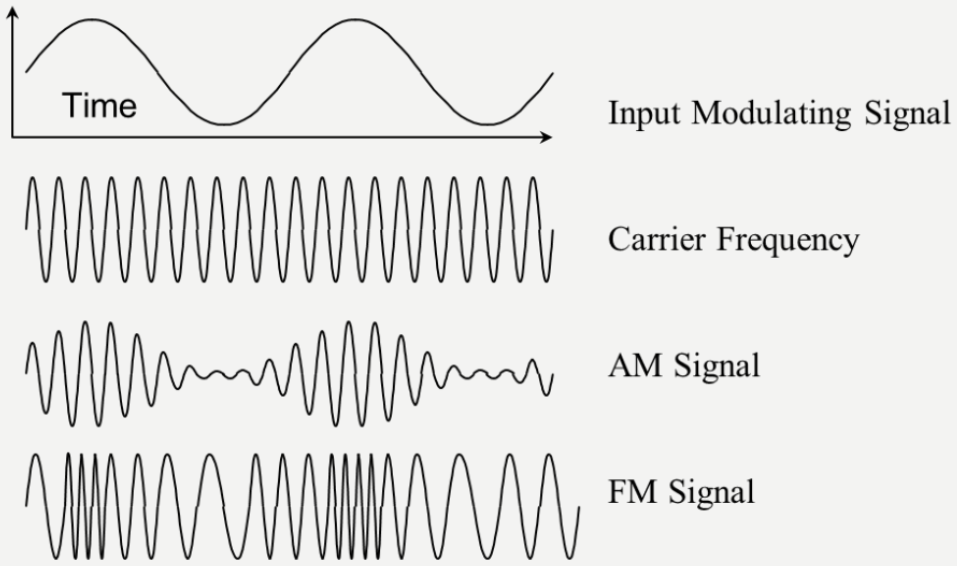
- Una modulazione è l'alterazione dell'ampiezza, della frequenza o della fase di un oscillatore provocata da un altro segnale

- L'oscillatore che viene modulato è detto portante (carrier), l'oscillatore che modula è detto modulante (modulator)

- In queste lezioni ci si soffermerà su 3 tipologie:
  1. Modulazione d'ampiezza
  2. Modulazione ad anello
  3. Modulazione di frequenza

TIPI DI MODULAZIONE: CONFRONTO GRAFICO

A scopo esemplificativo, mostriamo la modulazione di un segnale sinusoidale tramite un altro segnale sinusoidale



Si nota la variazione periodica rispettivamente del parametro di ampiezza (AM) e di frequenza (FM) nel segnale risultante

MODULAZIONE D'AMPIEZZA

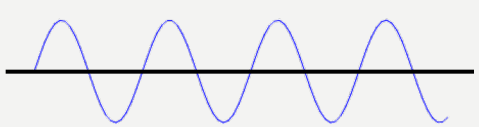
- La modulazione d'ampiezza è anche chiamata AM (da amplitude modulation); una sua variante è la modulazione ad anello, detta anche RM (ring modulation)

- Effetto: se la frequenza del segnale portante e del segnale modulante sono sufficientemente alte si ha la comparsa di frequenze nuove, che si aggiungono allo spettro della portante

- Tali frequenze vengono dette laterali, perché appaiono in modo simmetrico sopra e sotto la frequenza della portante

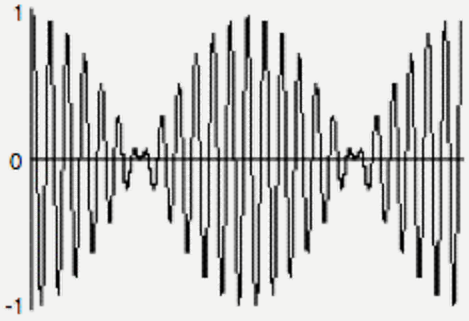
CONFRONTO TRA RM E AM

Segnale modulante:

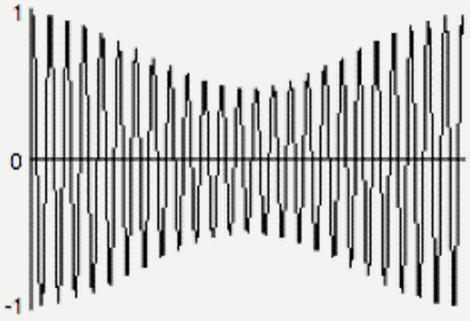


bipolare

unipolare



Ring Modulation (RM)



Amplitude Modulation (AM)

MODULAZIONE D'AMPIEZZA

- La modulazione d'ampiezza utilizza come segnale modulante un segnale unipolare, che si ottiene dal corrispondente segnale bipolare sommando una costante detta DC Offset (o componente di corrente continua)

- Tramite questa tecnica, si ottengono 3 frequenze:
  - la frequenza della portante f_p
  - la frequenza laterale $f_1 = f_p - f_m$
  - la frequenza laterale $f_2 = f_p + f_m$

- L'espressione matematica del segnale modulato in ampiezza è

$$V_{AM} = [A_p + A_m \cdot cos(2\pi f_m \cdot t)] \cdot cos(2\pi f_p \cdot t)$$

INDICE DI MODULAZIONE PER AM

- L'indice (o profondità) di modulazione è il rapporto tra le ampiezze di modulante e portante, e si esprime come:

$$I = A_m / A_p$$

- Si possono distinguere tre casi particolari:
  - $I = 0$ (in quanto $A_m = 0$) → si trasmette solo la portante non modulata in quanto è assente il segnale modulato
  - $I = 1$ (in quanto $A_m = A_p$) → la portante è modulata al 100 %. Gli inviluppi della parte positiva e di quella negativa del segnale si toccano in un punto e si è al limite della *distorsione*
  - $I > 1$ (in quanto $A_m > A_p$) → l'inviluppo non risulta più sinusoidale. Si verifica il taglio dei picchi del segnale modulato per la sovrapposizione degli inviluppi. E' la cosiddetta *sovramodulazione*

MODULAZIONE AD ANELLO

- La modulazione ad anello utilizza come segnale modulante un segnale bipolare, che oscilla tra valori positivi e negativi di ampiezza

- Se il segnale modulante è nullo, in uscita non vi sarà nemmeno la frequenza della portante
  - Questo non avviene per AM: anche se la modulante ha ampiezza 0, la portante assume come ampiezza il DC offset

- Altra differenza con AM: il risultato di una RM non contiene la portante, ma solo le frequenze laterali

- L'espressione matematica del segnale modulato ad anello è

$$V_{AM} = A_m \cdot \cos(2\pi f_m \cdot t) \cdot A_p \cos(2\pi f_p \cdot t)$$

MODULAZIONE DI FREQUENZA

- Nella modulazione di frequenza il segnale modulante bipolare viene sommato alla frequenza del segnale portante
  - Se $A_m = 0$, rimane solo l'oscillazione sinusoidale della portante
  - Se $A_m > 0$, quando l'uscita del modulante è positiva la frequenza modulata è più alta di quella base, quando è negativa rende la frequenza modulata più bassa di quella base

- Il massimo mutamento di frequenza che l'oscillatore portante subisce è detto deviazione di picco (*peak frequency deviation*)

- L'espressione matematica del segnale modulato in frequenza è

$$V_{FM} = A_p \cdot \cos[2\pi f_p \cdot t + A_m \cos(2\pi f_m \cdot t)]$$

SPETTRO DEL SEGNALE MODULATO CON FM

- Con la FM si genera una serie (teoricamente infinita) di coppie di bande laterali. Tali bande occorrono alle frequenze:

$f_p + f_m$	$f_p - f_m$
$f_p + 2f_m$	$f_p - 2f_m$
$f_p + 3f_m$	$f_p - 3f_m$
…	…

- Tanto maggiore è la deviazione di picco, tanto più l'energia si dispone sulle bande laterali e dunque queste assumono ampiezza sufficiente per poter essere udite

EFFETTI DI TREMOLO E VIBRATO

- Negli esempi visti finora, la frequenza dell'oscillatore modulante era sufficientemente alta da rientrare nel campo di udibilità

- Se la frequenza modulante è molto bassa, gli effetti percepiti non sono la comparsa di nuove frequenze, bensì:
  - nel caso di modulazione dell'ampiezza, una periodica lieve variazione che prende il nome di tremolo (valore soglia 10 Hz)
  - nel caso di modulazione della frequenza, una periodica lieve variazione che prende il nome di vibrato

IMPORTANZA STORICA

- Le tre tecniche di modulazione sono state originariamente utilizzate nella tecnica delle trasmissioni radio. L'applicazione alla sintesi del suono è stata ideata nello studio WDR di Colonia
  Tra i compositori più importanti: Karlheinz Stockhausen

- Questi procedimenti sono stati ampiamente adottati nella musica elettroacustica degli anni '50 e '60, con due finalità:
  - creare suoni complessi avendo a disposizione pochi oscillatori
  - operare modifiche timbriche di suoni tradizionali, anche dal vivo

ESEMPIO

(Modulations1.java)

- Il codice esemplifica le tecniche di modulazione di ampiezza e ad anello
- Con un opportuno passaggio parametri alla funzione che realizza la modulazione di ampiezza viene mostrato l'effetto di tremolo

ESERCIZIO

(Modulations2.java)

- Si arricchisca la classe di metodi statici Modulations presentata nell'esempio precedente con:
  - una funzione che implementi la modulazione di frequenza semplice
  - una funzione che implementi la modulazione di frequenza, e che consenta di passare in ingresso le frequenze fm1 e fm2, corrispondenti alla frequenza iniziale (all'istante 0) della modulante e alla sua frequenza finale (all'ultimo dei campioni generati)

- Si ricordi che per ottenere la modulazione di frequenza è necessario sommare i valori in uscita dell'oscillatore modulante alla frequenza base dell'oscillatore portante

SINTESI ADDITIVA

SINTESI ADDITIVA

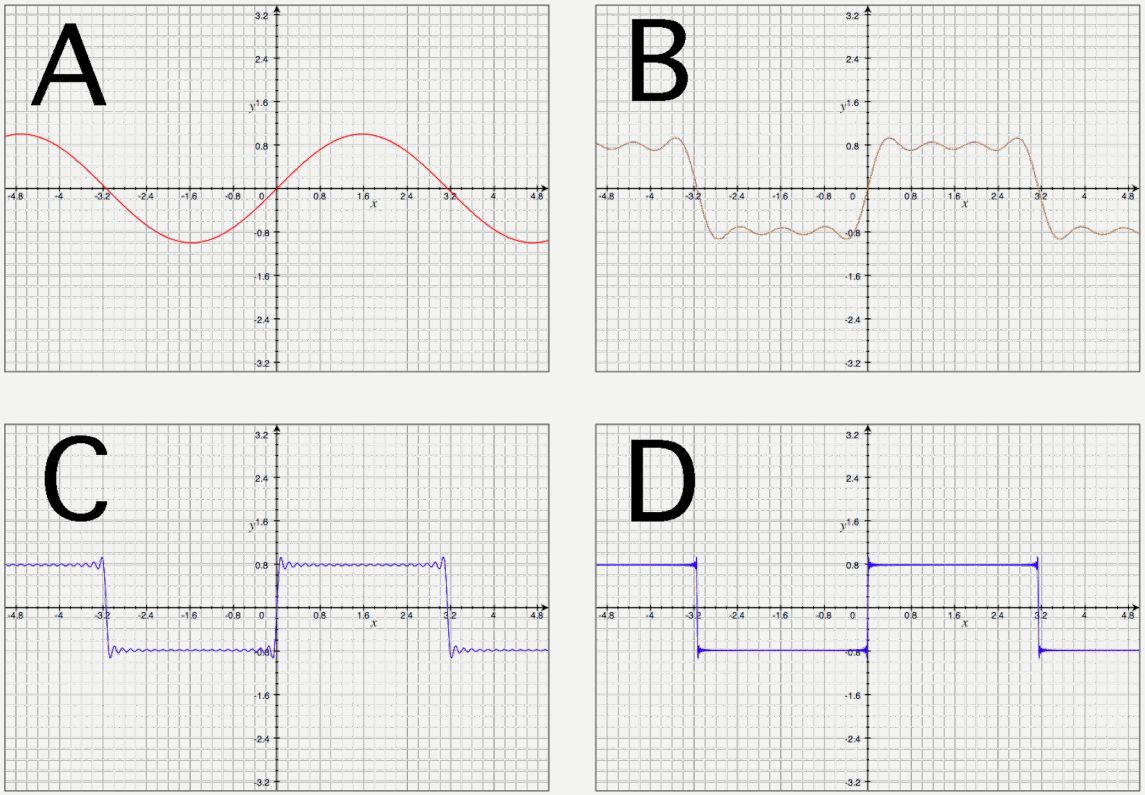
- La sintesi additiva in Fisica consiste nella somma di onde fondamentali con ottenimento di una risultante complessa

- In Acustica ed Elettronica/Informatica applicata al suono, la sintesi additiva è una tecnica che crea timbri, ossia forme d'onda potenzialmente complesse, sommando singole forme d'onda semplici, generalmente sinusoidali

- Il fondamento matematico della sintesi additiva è la serie di Fourier, che consente la rappresentazione di una funzione periodica mediante una combinazione lineare di funzioni sinusoidali. Questo tipo di decomposizione è alla base della cosiddetta analisi di Fourier

ESEMPIO: ONDA QUADRA

- Al crescere del numero di termini di una serie di Fourier presi in considerazione, migliora l'approssimazione dell'onda periodica
- La figura a lato mostra la somma di Fourier che approssima un'onda quadra al crescere del numero n di termini.
  In particolare:
  - n = 1 (A)
  - n = 6 (B)
  - n = 40 (C)
  - n = 200 (D)



OSSERVAZIONE

- Il teorema di Fourier afferma che qualsiasi funzione periodica può essere espressa come somma di sinusoidi; questo nel mondo reale non avviene mai

- Il teorema di Fourier però torna ad essere applicabile limitando l'analisi a porzioni sufficientemente brevi di suono

- Ad esempio, in generale gli strumenti musicali producono timbri complessi, ma con la caratteristica di essere suoni quasi periodici (si pensi alla fase di sustain)

CENNI STORICI SULLA SINTESI ADDITIVA IN MUSICA

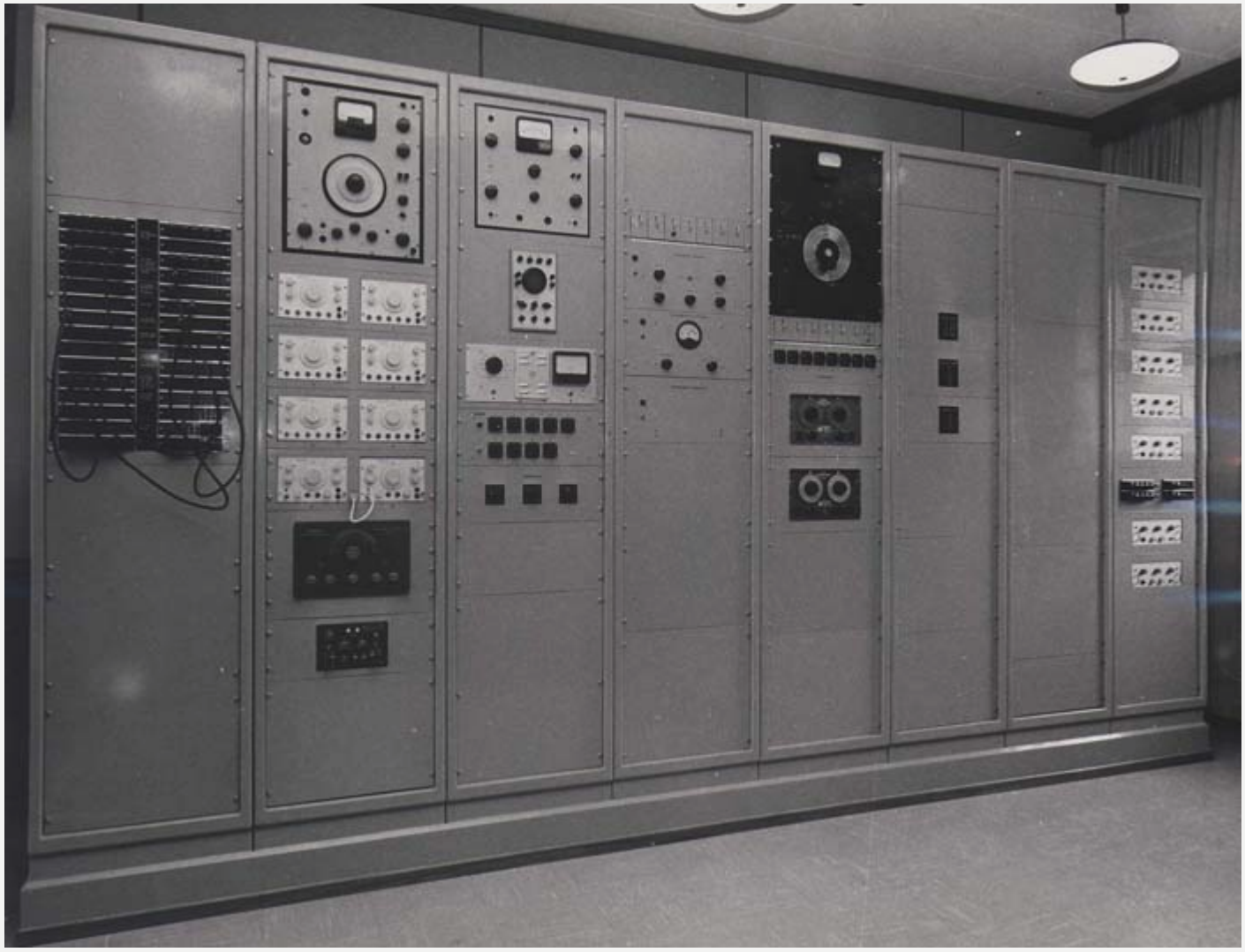
- Nella musica elettronica la sintesi additiva è stata storicamente la prima ad essere adottata

- Tra i primi sperimentatori di suoni elettronici costruiti con sintesi additiva, negli anni '50: Herbert Eimert e Karlheinz Stockhausen. In Italia tra i precursori: Bruno Maderna e Luciano Berio

- Esempio: K. Stockhausen, *Komposition 1953 Nr. 2 (Studie I)*
  https://www.youtube.com/watch?v=1GoUzk6fQAA

- Ai tempi il numero di oscillatori disponibili era molto esiguo.
  - Ad esempio, lo Studio di Fonologia della RAI di Milano (1955-1983), che era all'avanguardia a livello internazionale (assieme al NWDR - "Studio fur Elektronische Musik" di Colonia ed al Groupe de Recherche Musicale di Parigi), disponeva di un banco di 9 oscillatori. Era quindi possibile ascoltare in tempo reale il risultato della sovrapposizione di sole 9 sinusoidi

STUDIO DI FONOLOGIA RAI DI MILANO



SINTESI ADDITIVA A SPETTRO FISSO

- La sintesi additiva a spettro fisso si basa nella decomposizione di un suono periodico complesso in una somma di oscillazioni sinusoidali semplici che non variano nel tempo
  - In altri termini, i parametri degli oscillatori che concorrono alla formazione del timbro finale non variano nel tempo

- Il rapporto tra le frequenze degli oscillatori rispetto a una frequenza base può originare un valore intero (in tal caso si parla di rapporto armonico) o meno (rapporto inarmonico)
  - Molti strumenti musicali convenzionali hanno parziali prevalentemente armoniche, ma alcuni no (ad es. le campane)

ESEMPI
(Additive1.java)

- Somma di armoniche (multipli pesati della fondamentale)
- Onda a dente di sega approssimata
- Battimenti

SINTESI ADDITIVA A SPETTRO VARIABILE

- Si può ottenere varietà timbrica ad esempio rendendo variabile l'ampiezza delle componenti armoniche nel tempo

- Inoltre, non è necessario che le componenti siano (o rimangano) armoniche: per molti timbri le componenti inarmoniche sono di fondamentale importanza, e comunque possono aggiungere interesse a un dato timbro. Anche la variazione della frequenza degli oscillatori (ad esempio un passaggio da componenti inarmoniche ad armoniche) crea spettri variabili

ESERCIZIO
(Additive2.java)

- Aggiungere al codice degli esempi precedenti un metodo in cui il timbro sarà costituito dalla somma di 3 sinusoidi con frequenza variabile nel tempo rispetto alla frequenza passata f:
  - Sinusoide 1: da f a 1.1f
  - Sinusoide 2: da f a 1.5f
  - Sinusoide 3: da 2f a f