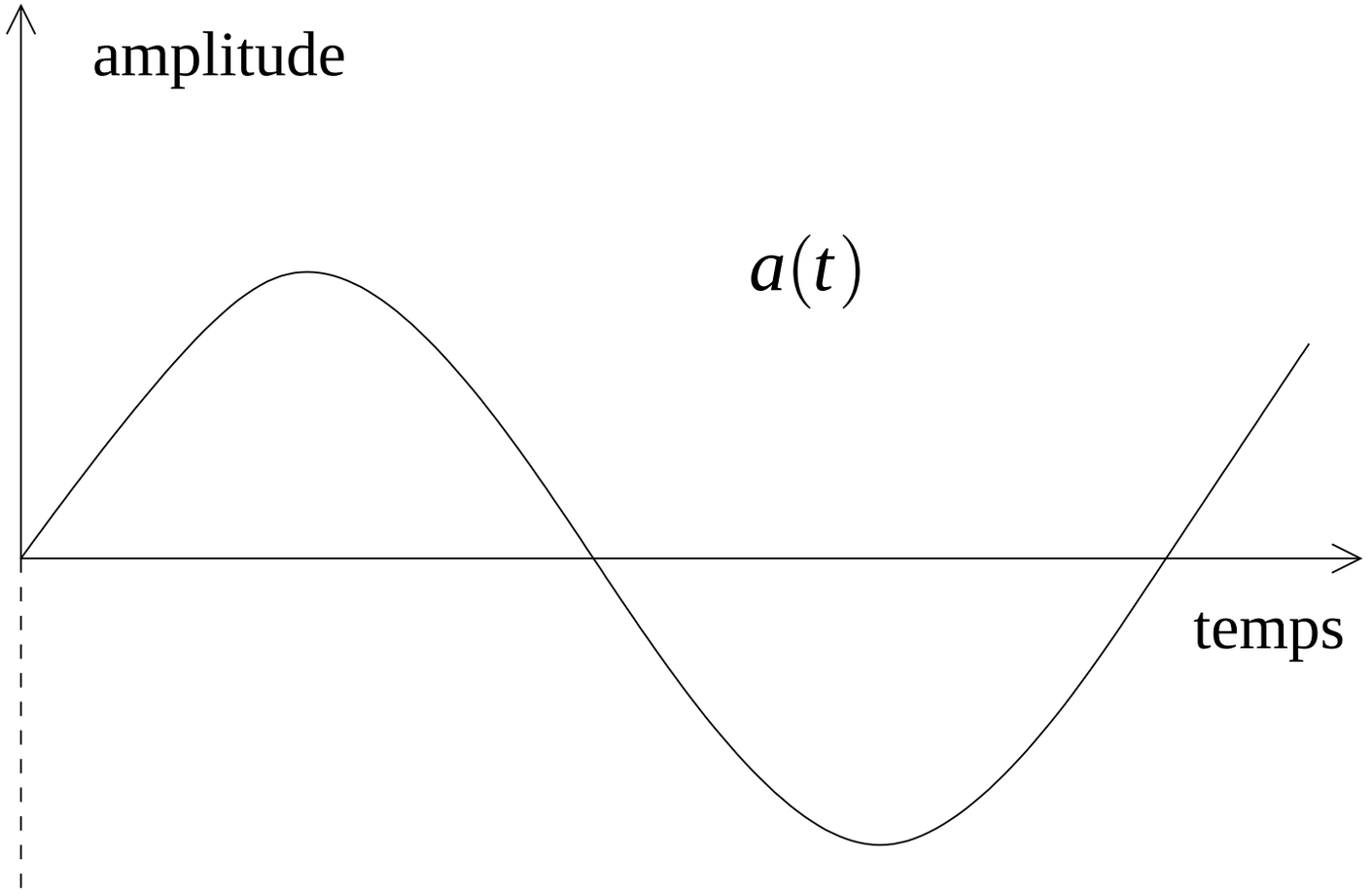
Formats Sonores

Matthias Robine

matthias.robine@labri.fr

Université Bordeaux

Représentation temporelle



$a(t)$: amplitude de l'onde en fonction du temps

(mesurée au niveau du tympan [oreille externe])

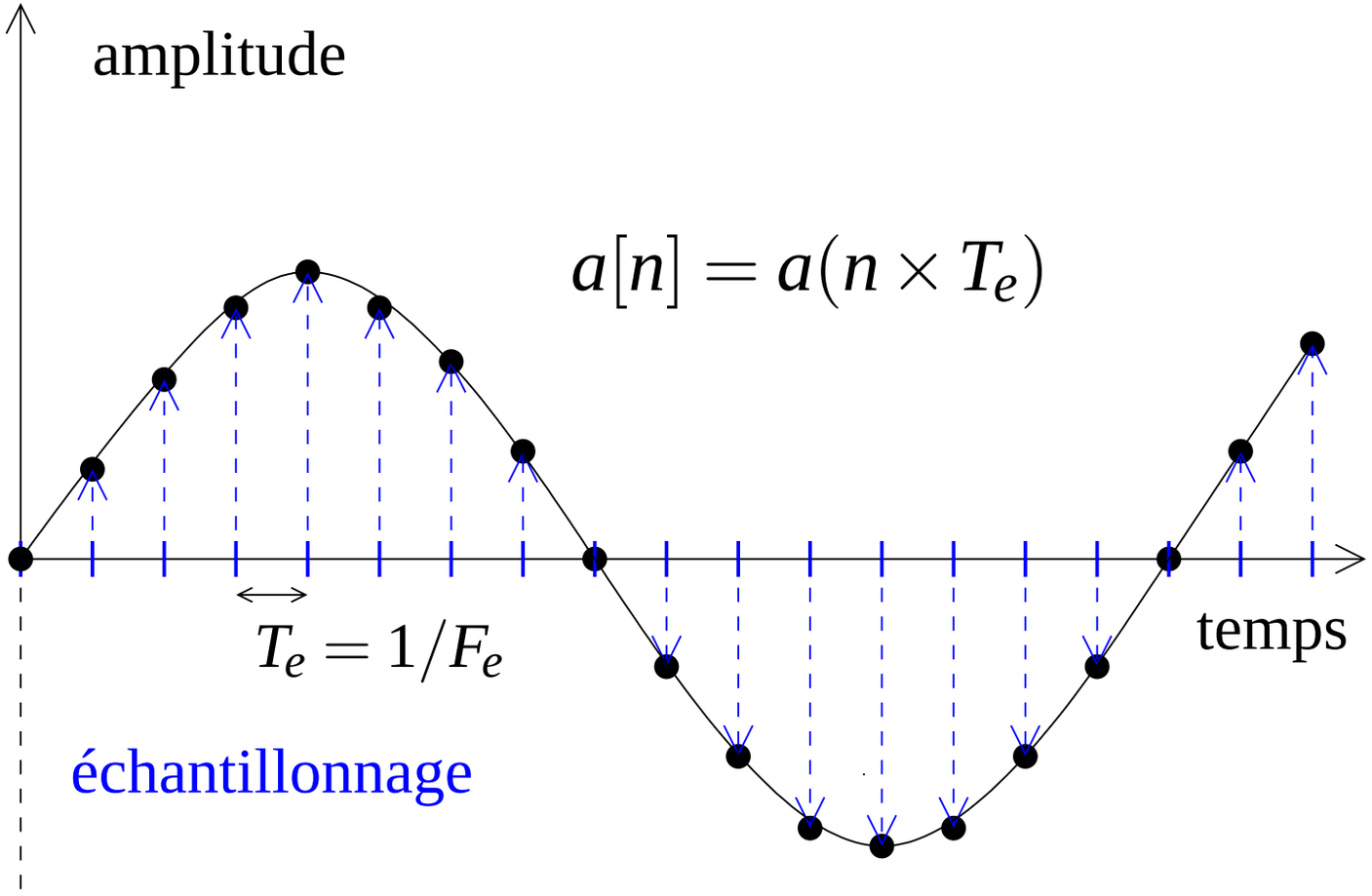
Son numérique

Utilisation d'un signal sonore dans un système informatique.
$\Longrightarrow$ conversion en un signal numérique (**numérisation**)

Ce procédé de numérisation est effectué en deux étapes :

- Échantillonnage
- Quantification

Échantillonnage



fréquence d'échantillonnage F_e (inverse de la période T_e)

Fréquences d'échantillonnage

- Parole : 8000 Hz
- radio FM numérique : 32000Hz
- CD audio : 44100Hz
- DAT : 48000 Hz

Musique professionnelle : multipistes numériques jusqu'à 96000 Hz
$\Longrightarrow$ CD audio : 44100 valeurs par secondes.

Échantillons

Question:

- Ecrire une fonction qui remplit un tableau de n réels `tab` donné en entrée par des échantillons représentant une sinusoïde de fréquence `freq`, d'amplitude `amp` et de phase `phase`, échantillonnée à la fréquence `R`.

Réponse:

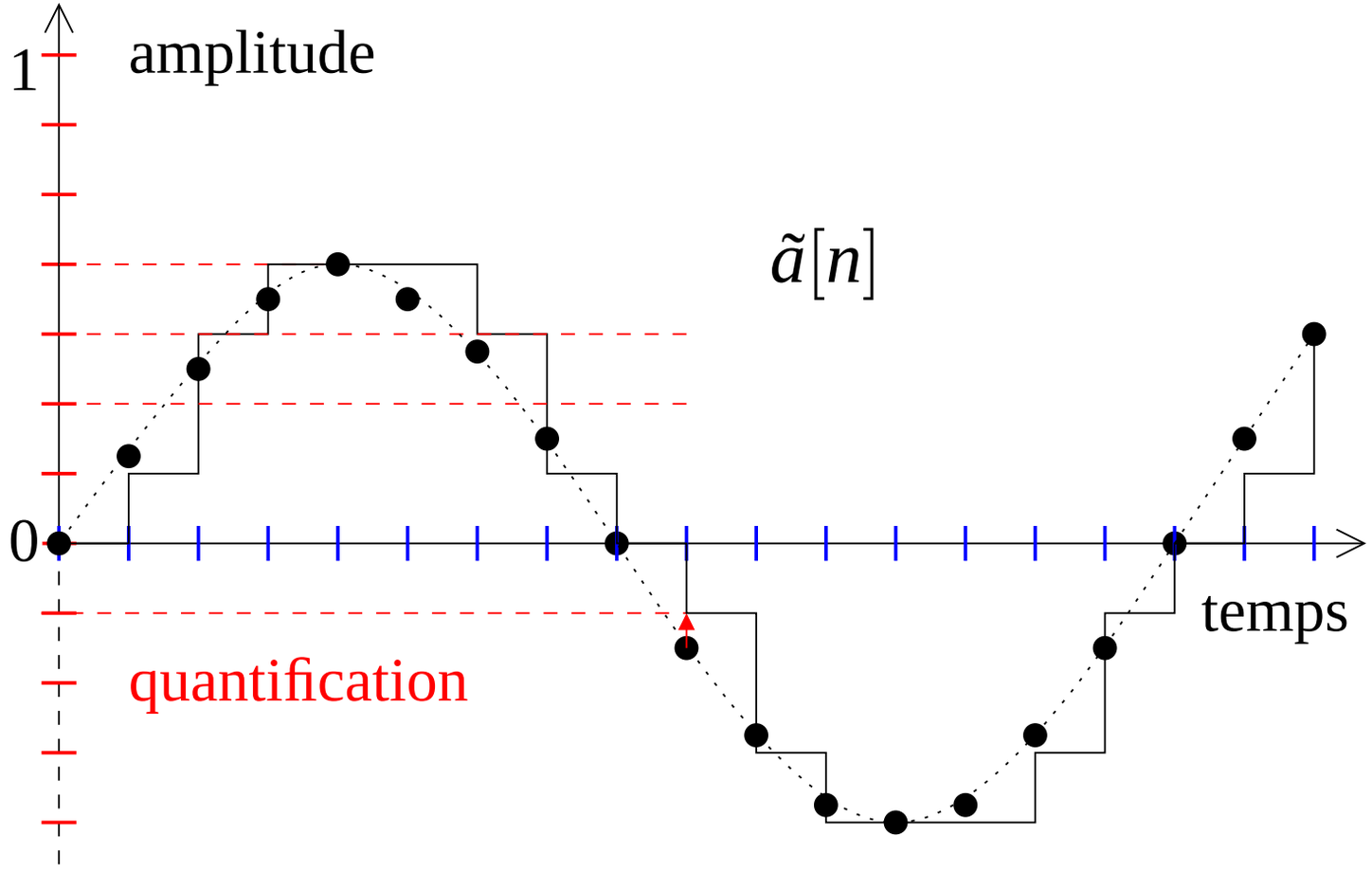
Quantification

- Échantillonnage : temps continu $\rightarrow$ temps discret
- Quantification : amplitude continue $\rightarrow$ amplitude discrète

La précision de cette étape de quantification est donnée par un **nombre de bits**

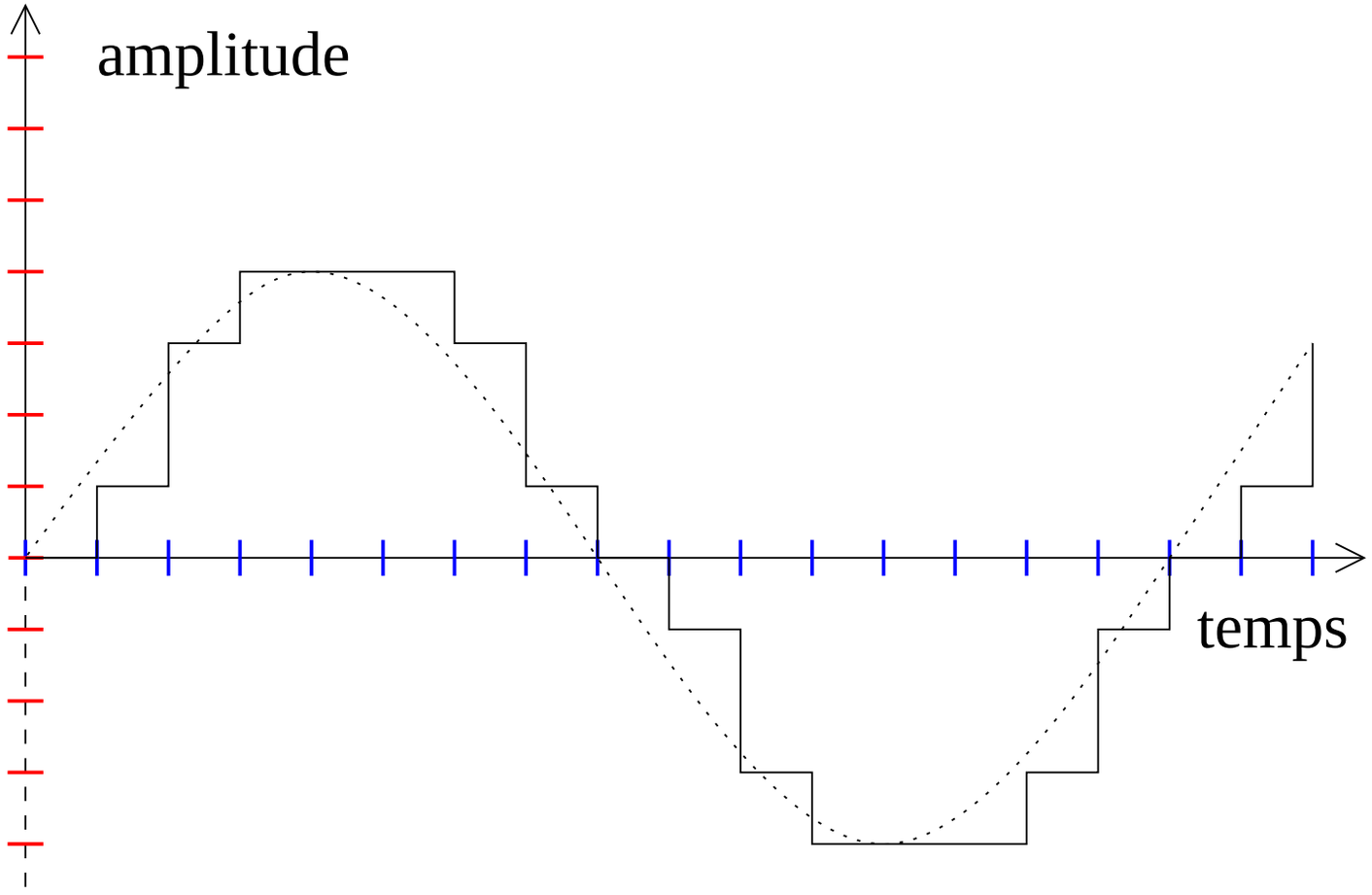
Ce nombre de bits indique le nombre de valeurs discrètes utilisées pour quantifier l'amplitude du signal analogique

Quantification



quantification sur k bits $\rightarrow 2^k$ valeurs possibles

Représentation discrète



discrétisation (numérisation) =
échantillonnage + quantification

[0, 1, 3, 4, 4, 4, 3, 1, 0, -1, -3, -4, -4, -4, -3, -1, 0, 1, 3]

→ *Pulse-Code Modulation* (PCM)

Représentation en machine

- Tableaux de valeurs d'échantillons
  - Entiers :
    échantillons entre -2^{BITS-1} et $2^{BITS-1}-1$

```
short son_short[] =
  { 0, 1, 3, 4, 4, 4, 3, 1, 0, -1, -3, -4, -4, -4, -3, -1, 0, 1, 3 };
```

  - Réels : **échantillons entre -1.0 et 1.0**

```
#define SIZE 19
#define BITS 4

float son_float[SIZE];

for (i=0; i<SIZE; i++)
  {
    son_float[i] = son_short[i] / pow(2, BITS-1);
  }
```

Représentation des échantillons

son **numérique** $\rightarrow$ nombres:

- arithmétique: entière / flottante
- calcul: non signé / signé
- bits: 8 / 16 / 24 / 32 / 64
- représentation machine: *little / big endian*

exemples:

- anciennes cartes son: entiers 8 bits non signés
- CD *(Compact Disc)*: entiers 16 bits signés *big-endian*
- DVD *(Digital Versatile Disc)*: entiers 24 bits signés

Représentations multi-canaux

Un son peut être représenté par plusieurs ondes

- diffusion mono/stéréo
- diffusion 5.1

$\Longrightarrow$ a un instant donnée, **une** valeur par canal
choix pour la polyphonie (ex: stéréo Gauche / Droite):

- séparé: G G $\cdots$ G D D $\cdots$ D
- **entrelacé**: G D G D $\cdots$ G D

Fréquences d’échantillonnage et taille

Question :
Combien d’octets sont nécessaires pour représenter un son d’une minute au format CD audio (44100Hz, 16 bits, mono) ?

Fréquences d'échantillonnage et taille

Tableau comparatif : taille des données / qualité numérisation
Taille des fichiers pour 1 min d'enregistrement.

	8 bits mono	8 bits stereo	16 bits mono	16 bits stereo
8000 Hz	420 Ko	900 Ko	900 Ko	1860 Ko
11025 Hz	600 Ko	1260 Ko	1260 Ko	2580 Ko
22050 Hz	1260 Ko	2580 Ko	2580 Ko	5160 Ko
44100 Hz	2580 Ko	5160 Ko	5160 Ko	10320 Ko

$\Longrightarrow$ CD audio : 1 minute $\approx$ 10 Mo

Formats de fichier

Son numérique sauvegardée sous la forme de fichiers sonores

Plusieurs types de format existent :

- format brut
- format structuré
- format compressé
- format musical
- format avec méta-données

Formats bruts

Les fichiers en format brut (`raw`) contiennent **uniquement** la suite des échantillons représentant le son

- `PCM` *(Pulse-Code Modulation)*
- `CDA` *(Compact Disc Audio)* *big-endian*
- `CDR` *(Compact Disc Raw)* *big-endian*
- “raw”...

Attention aux conversions : nécessaire de préciser les propriétés du fichier

Formats structurés

Formats bruts + entête (*header*) décrivant le fichier

- `WAV` (Waveform Audio File Format, dérivé du `RIFF`, Ressource Interchange File Format) : windows
- `AIFF` (Audio Interchange File Format) : mac
- `VOC` (Creative Voice) : parole
- `AU` : linux
- nombreux autres...

Formats compressés

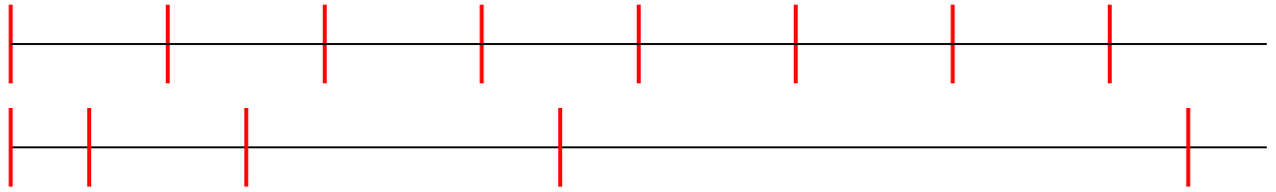
Compression non destructive (sans perte)
Exemple: `gzip`, `zip`, `bzip`, `...`

- fichier **identique** après compression/décompression
- taux de compression **faible**
- besoin du fichier **complet** avant compression

$\rightarrow$ Algorithmes généralistes, non spécifiques au son

Formats compressés

- Quantification logarithmique: codage μ-*law* AU, SND



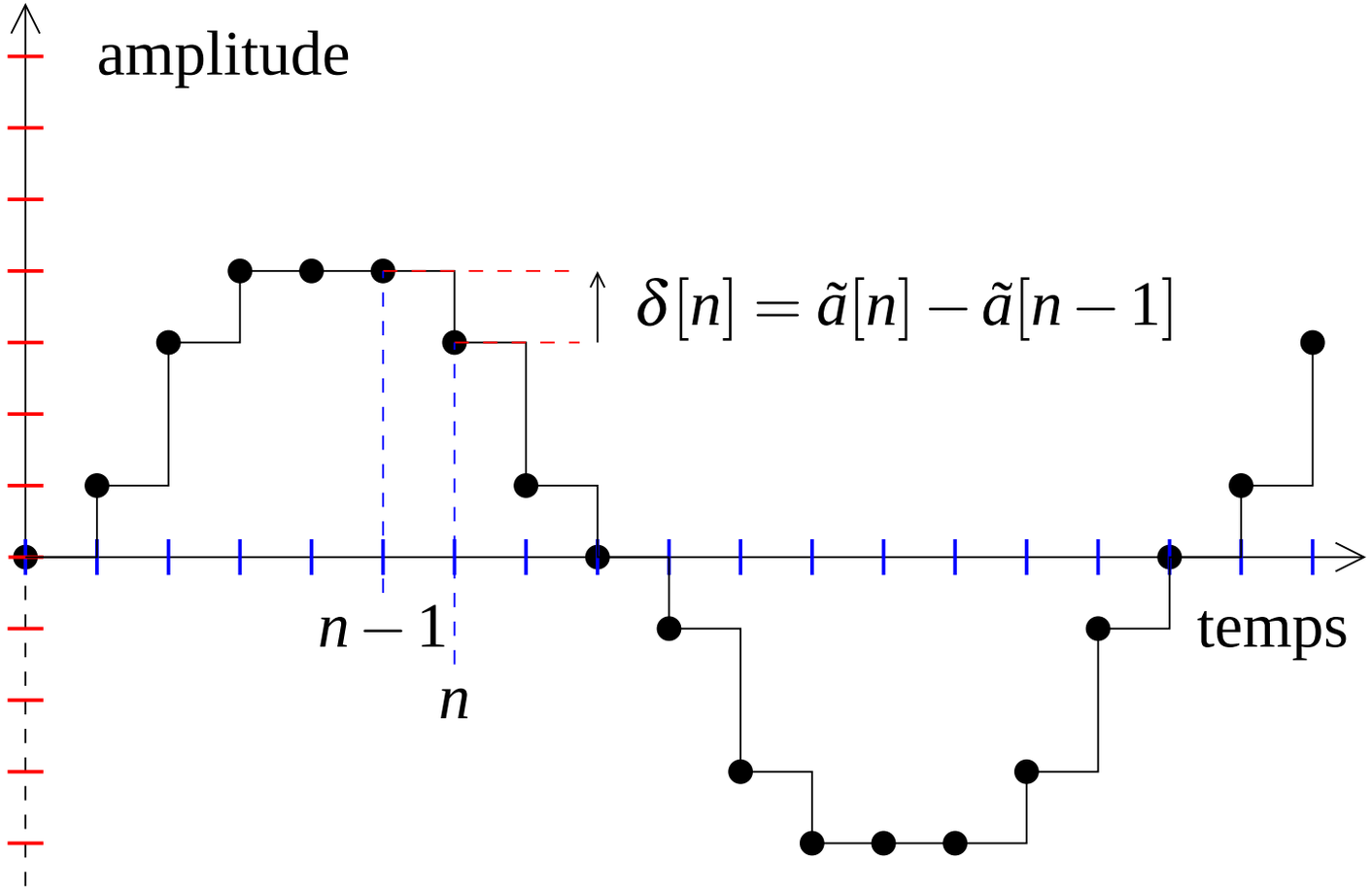
8 bits logarithmiques sont perceptivement équivalents à 12 bits linéaires

- ADPCM *(Adaptive-Delta Pulse-Code Modulation)*

au lieu de coder $\tilde{a}[n]$,
calcul du multi-ensemble des différences $\Delta = \bigcup_n \{\delta[n] = \tilde{a}[n] - \tilde{a}[n-1]\}$,
puis calcul l'histogramme des occurences de δ dans Δ,

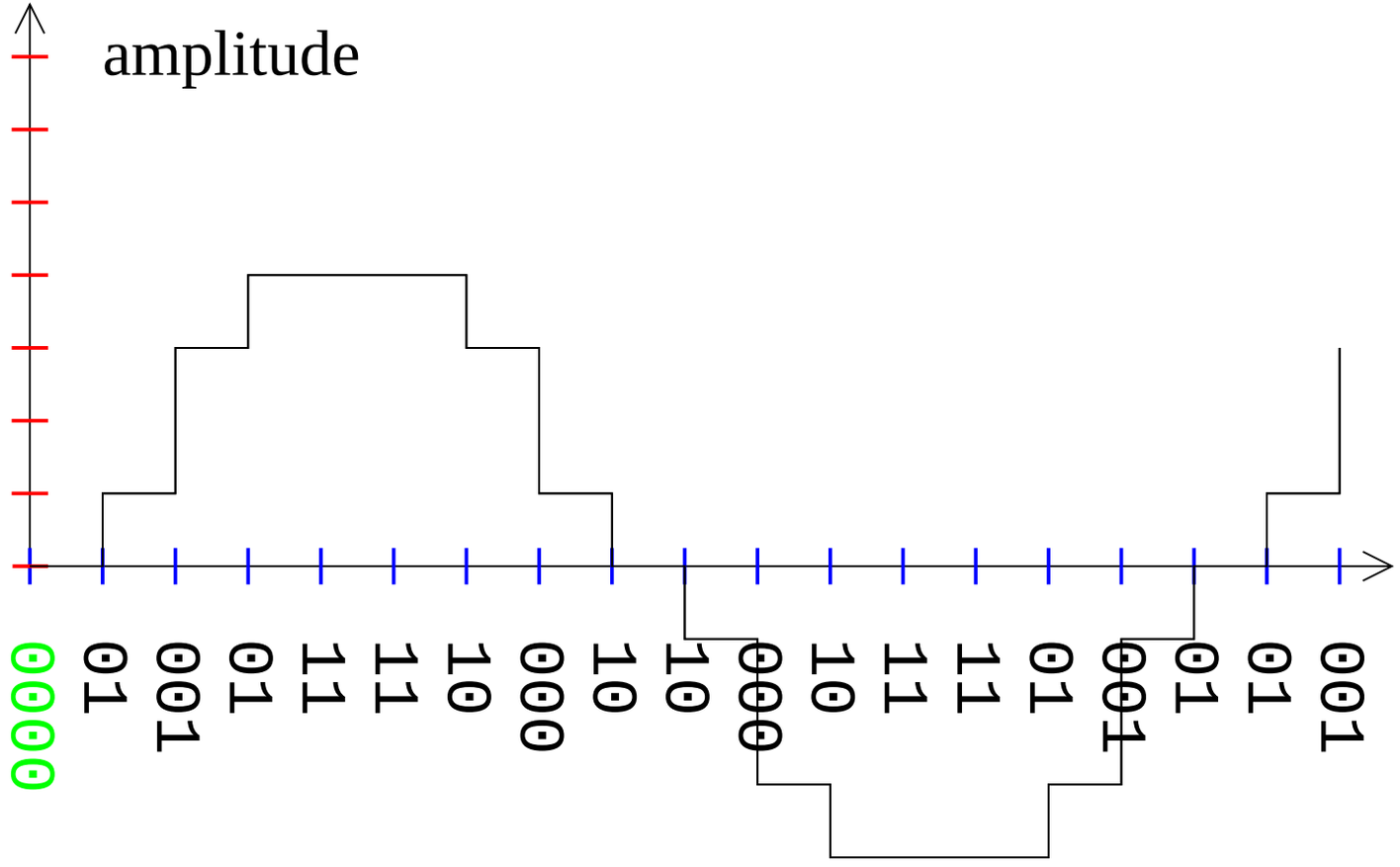
et codage de δ avec un nombre de bits utilisés inversement proportionnel au nombre de ses occurrences dans Δ (Huffman)

Codage ADPCM



$$\Delta = \{-2(2), -1(4), 0(4), 1(5), 2(3)\}$$

Comparatif PCM / ADPCM



$\rightarrow$ seulement 41 bits (au lieu de $19 \cdot 4 = 76$)

Formats compressés

Compression destructive

- compression spécifique au son
- algorithmes basés sur la perception
- suppression des informations non perçues

Principe : prise en compte

- des seuils limites d'audition
- des propriétés de *masquage temporel*
- des propriétés de *masquage fréquentiel*

Formats compressés

Exemples de formats de compression destructifs

- `MP3` : Motion Picture Expert Group, Audio Layer 3 (propriétaire) `www.mpeg.org`
- `OGG` Vorbis : **libre**
- `WMA` (Windows Media Audio) : windows
- `RA` (real audio)
- `VQF` : Yamaha, format abandonné
- MP3 Pro
- Dolby® AC3
- MPEG-4 AAC *(Advanced Audio Coding)*

Formats compressés

Attention aux compressions destructives

- transformations musicales : certaines informations supprimées peuvent être utiles après transformation
  exemple : allongement temporel

Principe général:

- analyse spectrale $\rightarrow$ domaine fréquentiel
- modèle psychoacoustique $\rightarrow$ degré de perceptibilité
- allocation de bits proportionnelle à la perceptibilité
- compression sans perte (Huffman)

Formats musicaux

Au lieu de se placer au niveau de l'onde sonore :

- pression acoustique en fonction du temps

Au niveau musical : événements musicaux (partition)

- hauteur des notes
- durée des notes
- . . .

Format principal : **MIDI** (Musical Instrument Digital Interface)

Formats de description

Format de description haut-niveau :

- Format d'échange entre applications son/musique `SDIF` (Sound Description Interchange Format)
- Description haut-niveau de contenu multimedia `MPEG7` (hauteur, tempo, . . .)

Applications son

- Utilitaire de conversion de format des fichiers son (`sox`)
- `file`: indique le type de fichier + informations

```
file son.wav
```

- Utilitaires de mixage (`aumix`, `xmix`, `...`)
- Lecteurs/enregistreurs de fichiers sons (`play`, `wavplay`)

```
play son.wav
```

```
play -d /dev/dsp1 son.wav
```

- Lecteurs de fichiers MIDI (`playmidi`, `timidity`)

Conversion de formats

Utilitaire sous **linux** : `sox` *(Sound eXchange)*

Options :

- fréquence d'échantillonnage `-r`
- nombre de bits `-sw` (pour *signed word*)
- nombre de canaux `-c`

Usage:

`sox` *source options destination*

Utilitaire `sox`

- Conversion de types de fichiers:

```
sox son.aiff son.wav
```

- Retrait de l'en-tête...

```
sox son.wav -r 44100 -c 1 -sw son.raw
```

- Rajout de l'en-tête...

```
sox -r 44100 -c 1 -sw son.raw son.wav
```

- Ré-échantillonnage:

```
sox son.wav -r 48000 son2.wav
```

Device **UNIX:** `/dev/dsp`

UNIX: principe du “tout est fichier”

- Fichiers spéciaux `/dev/dsp` et/ou `/dev/audio`
  (géré par le noyau UNIX,
  pas d’existence sur le disque dur)
- Commandes UNIX usuelles:
  `open`, `read` / `write`, `close`
- Lecture / écriture: le plus vite possible. . .
  - processus bloqué (endormi)
    quand le buffer est vide / plein
  - pas de véritable temps-réel:
    risque de clics

Lecture/écriture de son

Principe de la lecture d'échantillons :

- ouverture du fichier périphérique carte son `open`
- configuration des propriétés des échantillons `ioctl`
- écriture des échantillons dans le fichier `write`
- fermeture du fichier périphérique carte son `close`

Ouverture de la carte son

Ouverture du fichier `/dev/dsp` en mode écriture

```
fd_audio = open ("/dev/dsp", O_WRONLY);
if (fd_audio == -1)
  {
    perror("pbm ouverture /dev/dsp\n");
    exit(EXIT_FAILURE);
  }
```

Configuration du format de lecture

Utilisation de `ioctl` (*Input/Output Control*)

```
int ioctl(int d, int request, ...)
```

où d est un descripteur de fichier, `request` un numéro de requête et éventuellement une valeur...

Configuration du format de lecture

- `SOUND_PCM_SETFMT` : permet d'indiquer le format des échantillons (nombre de bits, little/big endian, signé/non signé)
- `SNDCTL_DSP_STEREO` : permet de préciser le nombre de voies
- `SNDCTL_DSP_SPEED` : permet d'indiquer la fréquence d'échantillonnage

- L'ordre des réglages est important
- Les paramètres peuvent être modifiés après l'appel
- `ioctl()` peut retourner une erreur...

Configuration du format de lecture

```
format = AFMT_S16_LE;
ioctl (fd_audio, SOUND_PCM_SETFMT, &format);
if (format != AFMT_S16_LE)  /* pbm format */
  { close (fd_audio); exit(EXIT_FAILURE); }

stereo = 1;
ioctl (fd_audio, SNDCTL_DSP_STEREO, &stereo);
/* la carte son de mon ordi est STEREO uniquement */
if (stereo != 1) /* pbm stereo */
  { close (fd_audio); exit(EXIT_FAILURE); }

speed = 44100;
ioctl (fd_audio, SNDCTL_DSP_SPEED, &speed);
if (speed != 44100) /* pbm Fech */
  { close (fd_audio); exit(EXIT_FAILURE); }
```

Écriture des échantillons

Écriture à l'aide du descripteur de fichier obtenu après ouverture de `/dev/dsp`

```
short *sample_buffer;
...
write(fd_audio, sample_buffer, size);
```

Fermeture de la carte son

Fermeture du descripteur de fichier correspondant à `/dev/dsp`

```
/* close dsp */
  close (audio);
```

Exemple: 1 minute de silence...

```
for (i=0; i<(60*speed); i++)
  {
    short samples[2] = { 0, 0 };

    write (audio, samples, 2*sizeof(short));
  }
```

TD

Développement de modules pour lire/écrire/modifier des fichiers `WAV`

- Module *son temporel*
- Lecture/écriture de fichiers `WAV`
- Affichage de la représentation temporelle d'un son
- Lecture d'un son

ATTENTION :

- Fichier `wav` : format `short` entiers sur 16 bits (compris entre -32767 et 32768)
- En mémoire : format `float` (ou `double`) : réels compris entre -1.0 et 1.0.

Étude du format WAV

Format très populaire

- Sous-ensemble du format `RIFF` (Ressource Interchange File Format) de Microsoft
- 2 entêtes (header) : format `RIFF` et `fmt`

Interchange File Format **(IFF)**

- Principe général (texte, son, image, vidéo, *etc*.)
- Standard indépendant
- Compatibilité totale :
  - Compatibilité ascendante:

    Les nouveaux logiciels doivent être capables de lire (**sans effort**) les anciens fichiers
  - Compatibilité **descendante**:

    Les anciens logiciels doivent être capables de lire aussi les nouveaux fichiers. . .

“Chunks”

- Solution adoptée: système entête *(header)* + *chunks*
- Structure d’un *chunk*:
  - type *(ID)*
  - taille *(size)*
  - données *(data)*
- Compatibilité:
  - chunks extensibles (versions ultérieures)
  - nouveaux chunks “optionnels”
- Algorithme:
  - *chunk* connu ?
  - non: l’ignorer, en passant au suivant (grâce à sa taille)
  - oui: selon que sa taille est
    - trop grande: lire les premières données et ignorer les autres
    - correcte: OK
    - trop petite: compléter avec des valeurs par défaut normalisées

Formats AIFF et RIFF Wave

- AIFF:
  Audio Interchange File Format
  - Apple + Commodore Amiga + SGI *(Silicon Graphics, Inc.)*
  - *big-endian* (Motorola, IBM, MIPS, *etc.*)
- RIFF WAVE:
  Resource Interchange File Format – Waveform
  - Microsoft
  - *little-endian* (Intel)

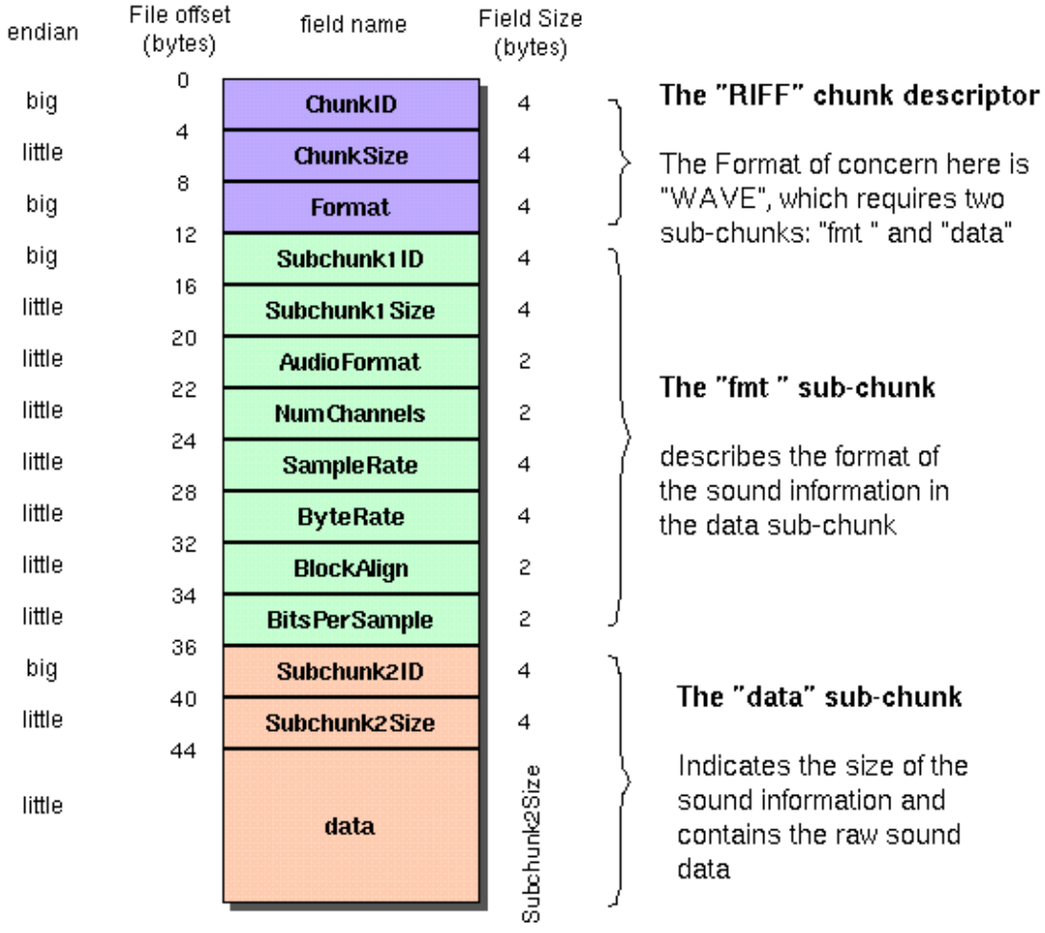
Entête WAV

Entête contient informations nécessaires, essentiellement :

- fréquence d'échantillonnage
- nombre de bits de quantification
- nombre de canaux
- taille du fichier

Format WAV

The Canonical WAVE file format

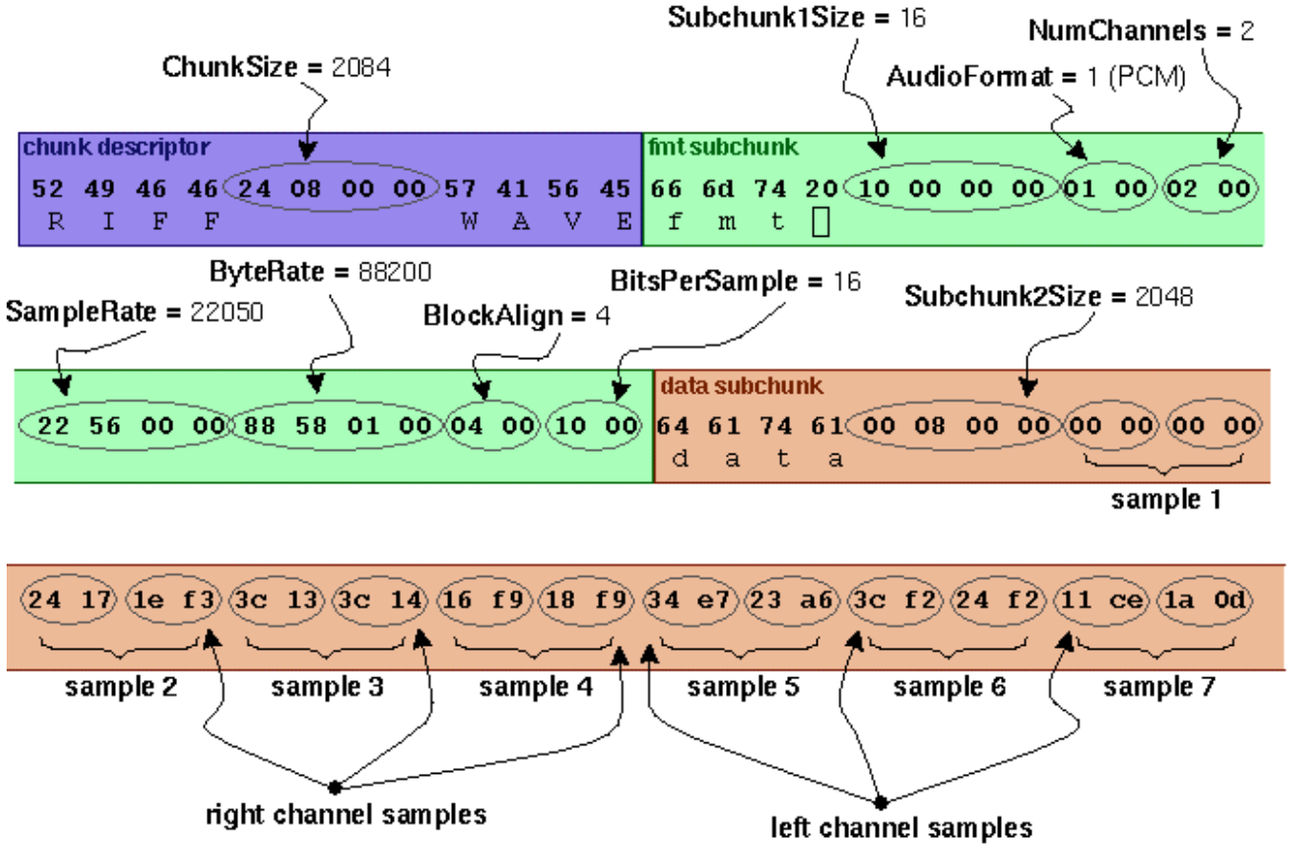


Format Wave: un exemple

champ	taille (octets)	type	valeur
name	4	texte	`RIFF`
size	4	entier	$4+(8+16)+(8+4n)$
type	4	texte	`WAVE`
name	4	texte	`fmt_` ($\leftarrow$ _ représentant un espace ' ')
size	4	entier	16
format	2	entier	1 (PCM)
channels	2	entier	2 (stéréo)
rate	4	entier	44100
bytes per second	4	entier	$rate \times channels \times \lceil bits/8 \rceil = rate \times 4$
block align	2	entier	2
bits	2	entier	16
name	4	texte	`data`
size	4	entier	$4n$
data	$2 \times 2 \times n$	entiers (16 bits)	$0\ 0 \cdots 0\ 0$

Données WAV

Après les 2 entêtes, données sous la forme d'échantillons (entrelacés) :



Données WAV

Bibliothèques audio sous Linux supportant les formats classiques

- `libaudiofile` : `http://www.68k.org/ michael/audiofile/`
- `libsndfile` : `http://www.mega-nerd.com/libsndfile/`