Codec newbies:
how to build a crappy lossless audio codec

https://codecs.multimedia.cx/2010/11/why-lossless-audio-codecs-generally-suck/

Kostya's Boring Codec World

Kostya's Rants around Multimedia

« Maybe the last word about Bink version b

The biggest curse in codec design »

Why Lossless Audio Codecs generally suck

Why there are so many lossless audio codecs? Mike, obviously, had his thoughts on that subject and I agree with my another friend who said: "it's just too easy to create lossless audio codec, that's why everybody creates his own".

Well, theory is simple: you remove redundancy from samples by predicting their values and code the residue. Coding is usually done with Rice codes or some combination of Rice codes and an additional coder — for zero runs or for finer coding of Rice codes. Prediction may be done in two major ways: FIR filters (some fixed prediction filters or LPC) or IIR filters (personally I call those "CPU eaters" for certain property of codecs using it). And of course they always invent their own container (I think in most cases that's because they are too stupid to implement even minimal support for some existing container or even to think how to fit it into one).

Let's iterate through the list of better-known lossless audio codecs.

1. ALAC (by Apple) — nothing remarkable, they just needed to fit something like FLAC into MOV so their players can handle it
2. Bonk— one of the first lossless/lossy codecs, nobody cares about it anymore. Some FFmpeg developers had intent to enhance it but nothing substantial has been done. You can still find that "effort" as Sonic codec in `libavcodec`.
3. DTS-HD MA — it may employ both FIR and IIR prediction and uses Rice codes but they totally screwed bitstream format. Not

What is sound?

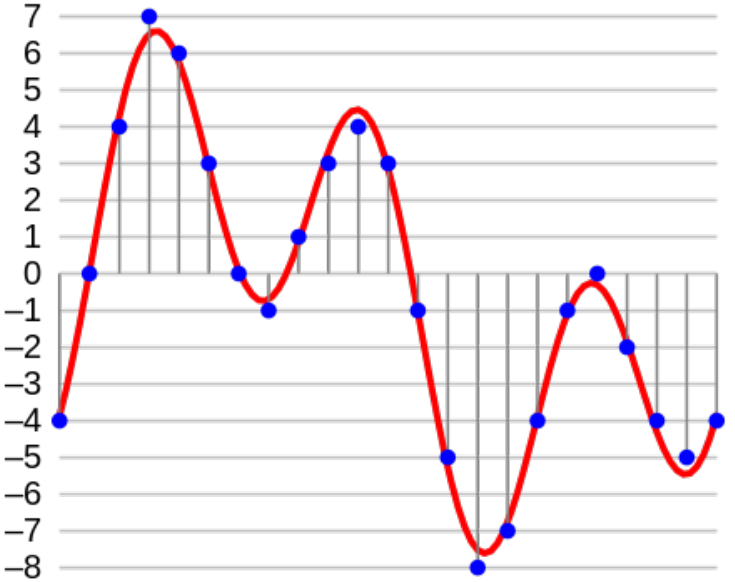
Audible wave of pressure, through a transmission medium (air)

In a computer?

- Must be quantized to be stored inside a computer.

  Linear **Pulse-Code Modulation** (PCM)

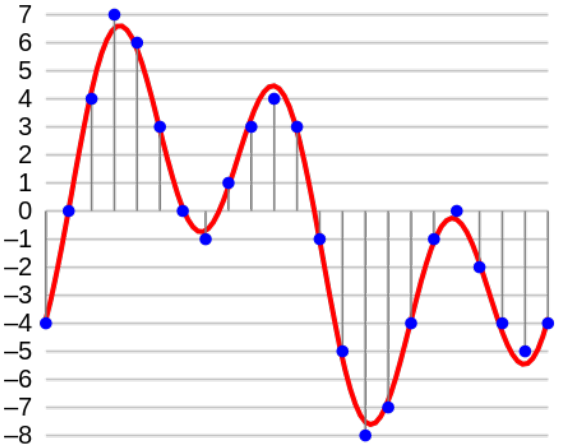
- Sampling frequency: number of values per second
  (ex: 44100 Hz, 48000Hz)



Storing sound

- As an array:
  - Of integer values
    (16 bit, 32 bit ...)
    ex: [-4, 0, 4, 7, 6, 3 …]
  - Or float values
    (32 bit)
- Uncompressed format:

  ex: Microsoft Wave Format



But this take space…

- Let’s compute the size!
- Track to encode:
  - 3 minutes songs,
  - Stereo: 2 channels,
  - Sampling: 44100 Hz,
  - Format: 16 bit integer.

- Size:

  3 * 60 * 2 * 44100 * 2

  = 30.28 MB !

So...

- Compression can be useful!
  - Lossy
    - Decompression do not restore the original discretized wave.
    - Tradeoff between data size and quality.
    - Ex: MP3, AAC…
  - Lossless
    - Decompression does restore the original discretized wave.
    - FLAC, ALAC, ALS...

CLAC : Crappy Lossless Audio Codec

Cut the input wave into frames

- For random access.

  Decompress them independently

- Optimize the compression parameters to each frame.
- Example frame sizes: 512, 1024, 2048...

Any idea?

Compression is prediction

Compression is prediction

- Guess what the next data will be!
- Encode the **difference** between the **prediction** and the **signal**.
- Predictor accurate most of the time,

  → small prediction errors,

  → few bits needed to encode them.
- Challenges:

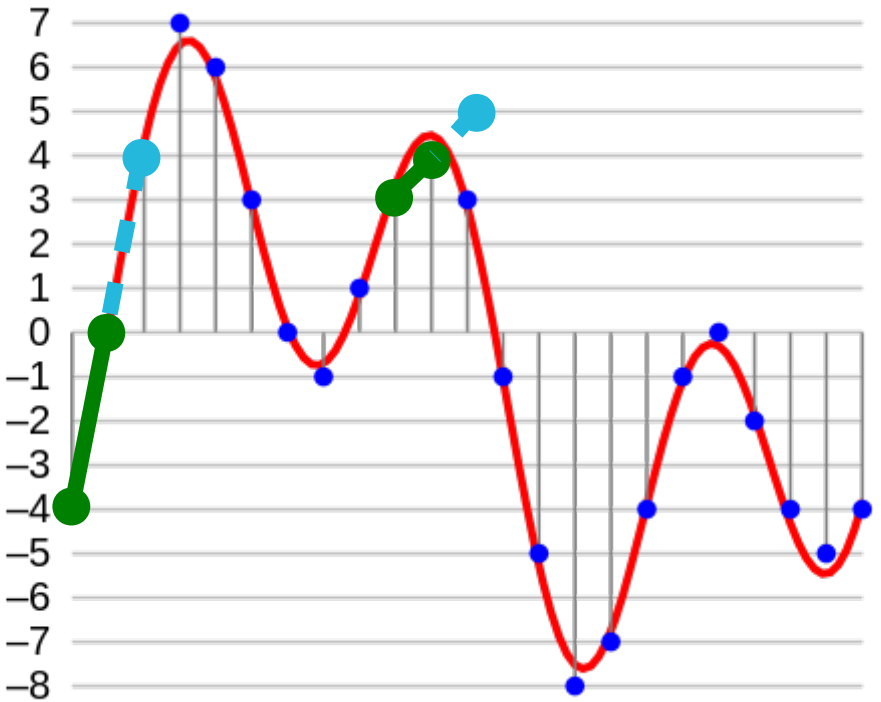
  1. Find the best predictor.
  2. Encode efficiently the prediction error.

1. Find the best predictor

- Linear predictor
  - **p(t) = s(t – 1)**
  - **p(t) = 2 * s(t – 1) – s(t – 2)**

    p(2) = 2 * s(1) – s(0)

    p(2) = 2 * -4 – 0 = 4
  - LPC...
- How to find the best predictor ?



Linear predictive coding (LPC)

- Widely used for lossless sound coding (Shorten, FLAC, MPEG-4 ALS...)
- Basic idea: current value can be closely approximated as a linear combination of past samples.

$$p(t) = \sum_{k=1}^{o} \alpha_k s(t-k)$$

p(t): predicted value
s(t): wave signal
o: predictor order

- How the get the α_k values?
  - Levinson-Durbin algorithm,
  - Must encode the α_k in the compressed signal (actually partial coefficients).
- How to choose the order?

Measure predictor efficiency

- X: the prediction signal to compress (n symbols)
- Compute
  - The compressed signal
    - The most accurate but slow
  - Shannon entropy
    - “Average amount of information produced by a probabilistic stochastic source of data” (Wikipedia)
    - $E(X)=\sum_{i=1}^{n} P(x_i)\log_2(P(x_i))$ $\quad e(X)=E(X)\times n$
    - Theoretical limit of the signal compression rate
    - Good predictor but slow
  - Flac encoder method ???
    - $S(X)=\sum_{i=1}^{n} x_i$ $\quad e(X)=\frac{1}{2}\log_2(\frac{1}{2}\times\frac{S(X)}{n})\times n$

2. Encode the prediction error

- Entropy coding
  - Lossless data compression scheme
  - Replace fixed length input symbol with a corresponding variable-length output codeword.
  - Aim: frequent symbols have a small size codeword.
  - Huffman coding
  - Arithmetic coding
    - Range coding

Range coding example

- Encode a symbol sequence with a very big integer.
- Example (from Wikipedia):
  - Thee symbols: A, B, X (end of message).
  - AABAX (10^5 combinations)
  - Symbol probabilities:
    - A: 0.60
    - B: 0.20
    - X: 0.20

Range coding example

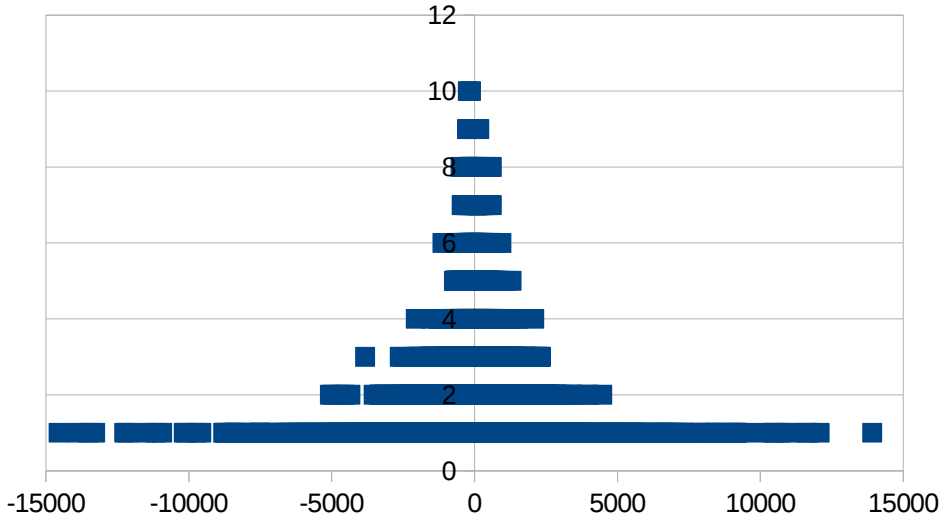
- Initial range: [0, 100000]
- 1. Symbol A
  - A: [0, 60000)
  - B: [60000, 80000)
  - X: [80000, 100000)
- 2. Symbol A
  - AA: [0, 36000)
  - AB: [36000, 48000)
  - AX: [48000, 60000)
- 3. Symbol B
  - AAA: [0, 21600)
  - AAB: [21600, 28800)
  - AAX: [28800, 36000)
- 4. Symbol A
  - AABA: [21600, 25920)
  - AABB: [25920, 27360)
  - AABX: [27360, 28800)

Range coding example

- 5. Symbol X
  - AABAA: [21600, 24192)
  - AABAB: [24192, 25056)
  - AABAX: [25056, 25920)
- “251”: 3 digits prefix than fit in our final range
- How to select a big enough initial range?
  - Does not matter.
  - Remove from our currant range the left-most digits that won’t change.

Probability distribution

- The range encoder/decoder need a symbol distribution.



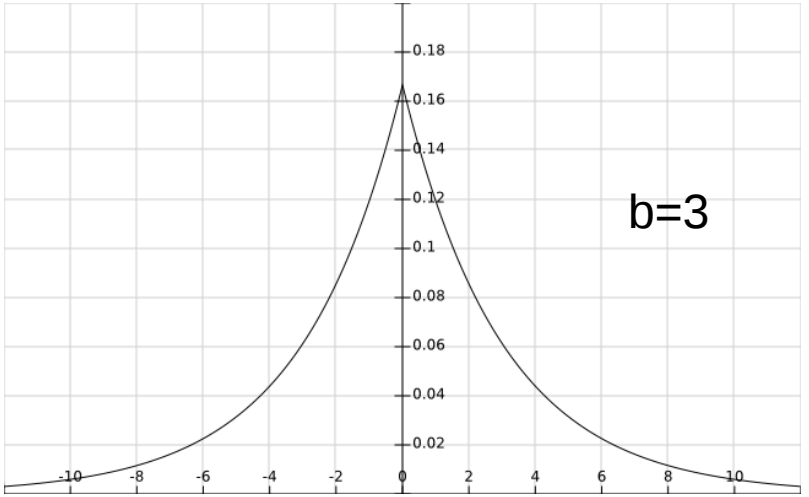
ex. of LPC distribution

- How to transmit it to the decoder?
  - Save it in the stream?
    - Way too big!
  - Estimate it with few parameters
    - Transmit them in the stream.
- Alternative method: use an adaptive distribution

Finding distribution parameters

- Laplace distribution

$$PDF = \frac{1}{2b} \mathrm{e}^{-\frac{|x|}{b}}$$



- Parameter estimation

$$\hat{b} = \sum_{i=1}^{n} |x_i|$$

- Transmit b value in the compressed data

Demo

Remarks

- CLAC is very limited:
  - 16 bit integer PCM,
  - Does not support multi-channel,
  - Does not store meta-data,
  - Has a single predictor,
  - Is not secure.
- CLAC is slooooowwwwww…
  - Not optimized at all

And remember…

Compression is prediction

Thank you!

Bibliography

- Robinson, Tony. "SHORTEN: Simple lossless and near-lossless waveform compression." (1994).
- https://en.wikipedia.org/wiki/Range_encoding
- https://en.wikipedia.org/wiki/Laplace_distribution
- Liebchen, Tilman, et al. "The MPEG-4 Audio Lossless Coding (ALS) standard-technology and applications." Proc. 119th AES Conv. 2005.