TRƯỜNG ĐẠI HỌC XÂY DỰNG HÀ NỘI
KHOA CÔNG NGHỆ THÔNG TIN

LẬP TRÌNH ÂM THANH

CHƯƠNG 3

Số hoá âm thanh

Mô-đun M2 · 5 tiết · CLO1 · Mã học phần 531833 · 75 tiết · Học kỳ 5

Hà Nội – 2026

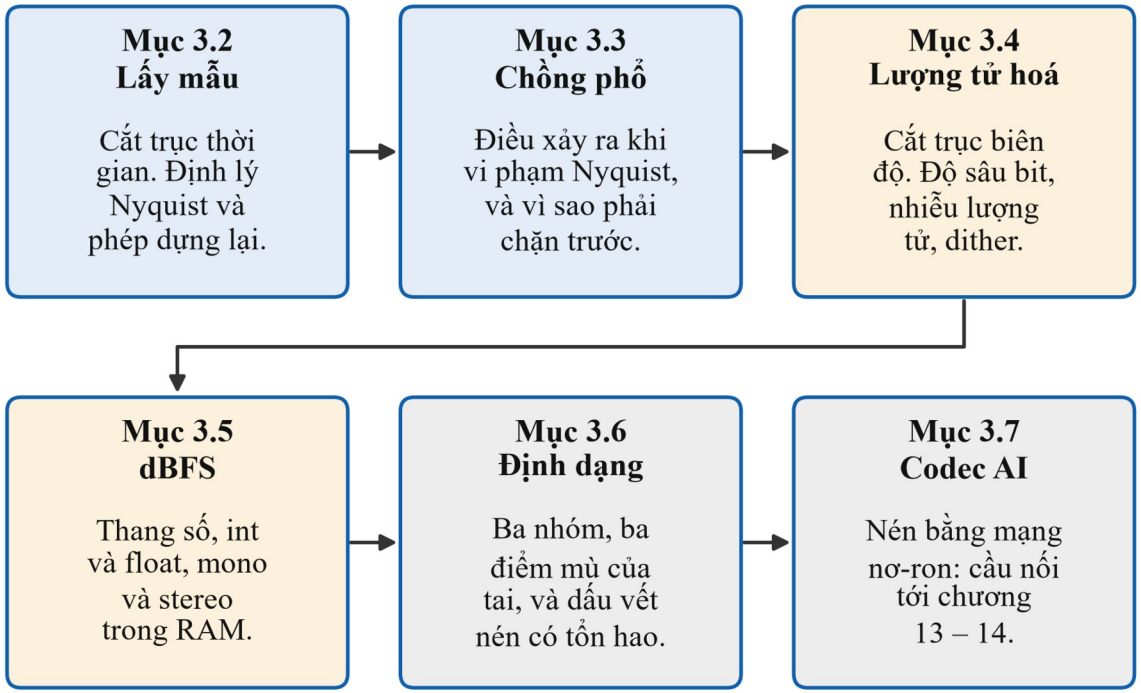
Sau chương này người học phải làm được

- Phân biệt hai phép rời rạc hoá: mỗi phép cắt trục nào, do tham số nào điều khiển.
- Phát biểu, vận dụng định lý Nyquist–Shannon; giải thích vì sao dãy mẫu chứa đủ thông tin.
- Tính tần số thấy được của tần số vượt Nyquist; hiểu vì sao phải chặn chồng phổ trước khi lấy mẫu.
- Nêu quan hệ độ sâu bit – số mức – dải động; hiểu dither làm gì và dùng ở bước nào.
- Đổi giữa biên độ mẫu và dBFS (decibel so với mức toàn thang, Full Scale); phát biểu nguyên tắc cốt lõi về số nguyên và số thực.
- Phân biệt ba nhóm định dạng, ba điểm mù của tai và dấu vết nén có tổn hao trên phổ.
- Mô tả nguyên lý codec nơ-ron và mối nối với AI (Artificial Intelligence – trí tuệ nhân tạo) tạo sinh ở cuối học phần.

MỤC 3.1

Vị trí chương học trong học phần

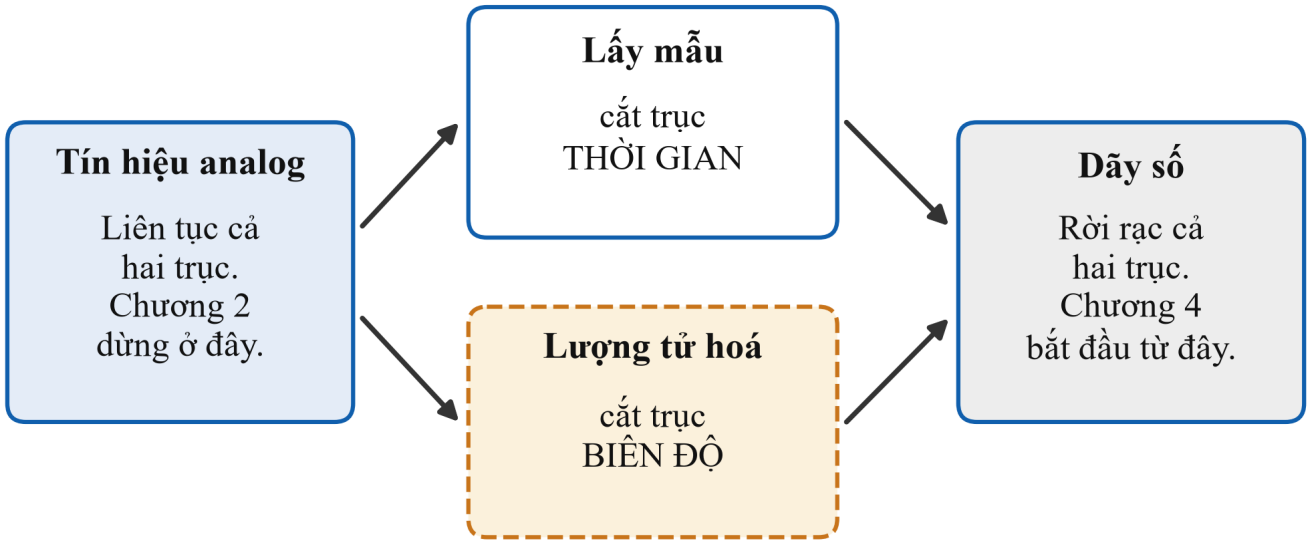
- Mở ô vuông ADC (Analog-to-Digital Converter – bộ chuyển đổi tương tự sang số) của chương 2: bên trong chỉ có hai phép.
- Chương bản lề: từ chương 4, mọi dòng mã làm việc trên dãy số chương này tạo ra.
- Quy ước nhỏ ở đây (khoảng mẫu, mono/stereo, mốc dBFS) xuất hiện ở mọi bài tập về sau.



Chương 2 dừng ngay trước khối ADC. Chương 3 mở khối ấy ra xem bên trong. Bên trong chỉ có hai phép: cắt trục thời gian và cắt trục biên độ. Mọi thứ còn lại của chương này là hệ quả của hai phép ấy; từ chương 4, mọi thứ diễn ra trên dãy số.

Số hoá gồm hai phép độc lập

- Lấy mẫu cắt trục thời gian; lượng tử hoá cắt trục biên độ.
- Hai bộ phận khác nhau thực hiện, gây hai loại sai số khác nhau.
- Điều khiển bằng hai tham số: tần số lấy mẫu và độ sâu bit.
- Tách được hai phép thì nội dung chương dễ theo dõi.



Hai phép độc lập nhau: do hai bộ phận khác nhau thực hiện, gây hai loại sai số khác nhau, kiểm soát bằng hai tham số khác nhau.

Lấy mẫu do TẦN SỐ LẤY MẪU điều khiển; lượng tử hoá do ĐỘ SÂU BIT điều khiển.

Hình 3.1 — Hai phép rời rạc hoá độc lập: lấy mẫu cắt trục thời gian, lượng tử hoá cắt trục biên độ

Ba tham số của một tệp âm thanh số

Tham số	Ký hiệu quen	Do phép nào sinh ra	Học ở mục
Tần số lấy mẫu	sr, fs	Phép cắt trục thời gian	3.2
Độ sâu bit	bits, b	Phép cắt trục biên độ	3.4
Số kênh	ch, channels	Quy ước lưu trữ	3.5.3

Ba con số này quyết định dung lượng, dải tần biểu diễn, dải động và hình dạng mảng trong bộ nhớ.

MỤC 3.2

Lấy mẫu và định lý Nyquist

- Lấy mẫu: đo giá trị tín hiệu tại những thời điểm cách đều nhau.
- Chỉ dãy chấm được lưu, đường liền nét thì không.
- Câu hỏi của mục: dựng lại đường liền từ dãy chấm được không, và với điều kiện nào?

Tần số và chu kỳ lấy mẫu

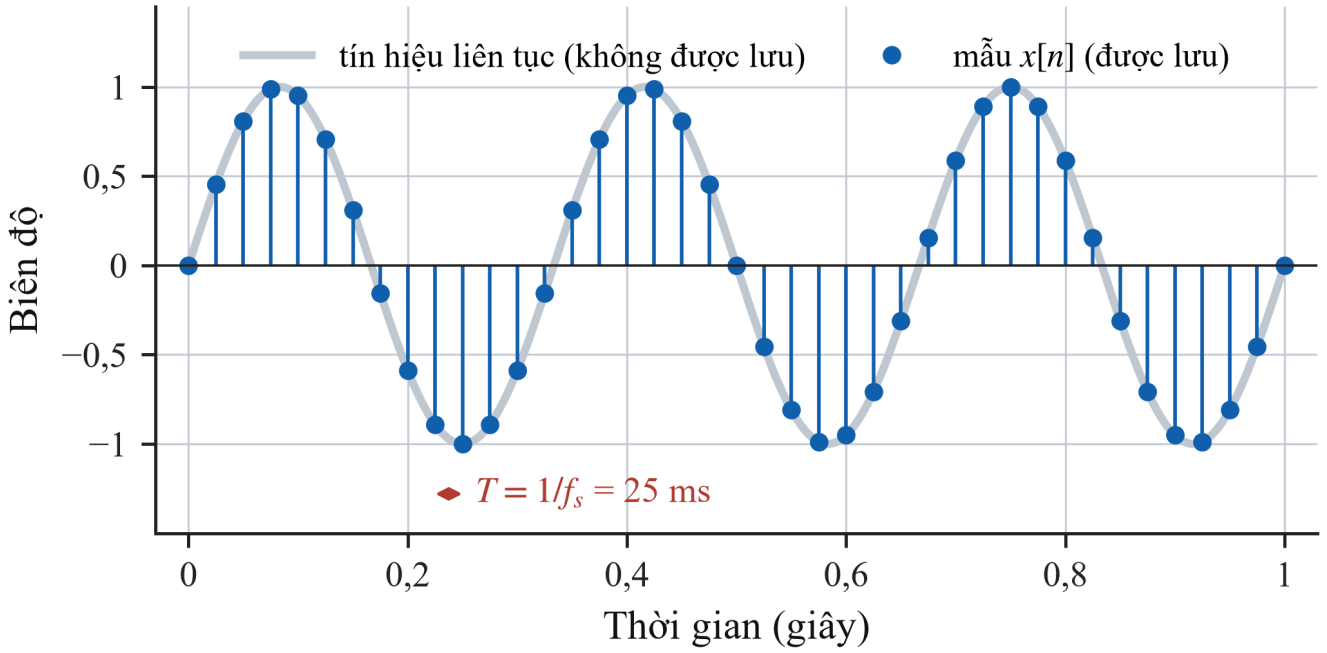
$$x[n] = x(nT), \qquad T = \frac{1}{f_s}$$

(3.1)

- x[n] là mẫu thứ n, đo tại thời điểm nT.
- fs là số lần đo mỗi giây (đơn vị Hz); T = 1/fs là chu kỳ lấy mẫu.
- Chuẩn CD 44.100 Hz: hai mẫu cách nhau 22,7 µs.
- Bài hát 4 phút stereo có khoảng 21,2 triệu con số.

Chỉ dãy chấm được lưu lại

- Đường liền nét không được lưu ở đâu cả.
- Máy tính chỉ biết dãy số: 44.100 con số mỗi giây cho mỗi kênh.
- Từ dãy chấm ấy, đường liền có dựng lại được không?
- Định lý Nyquist–Shannon trả lời câu hỏi này.



Hình 3.3 — Lấy mẫu: chỉ dãy chấm được lưu lại, đường liền nét thì không

Định lý lấy mẫu

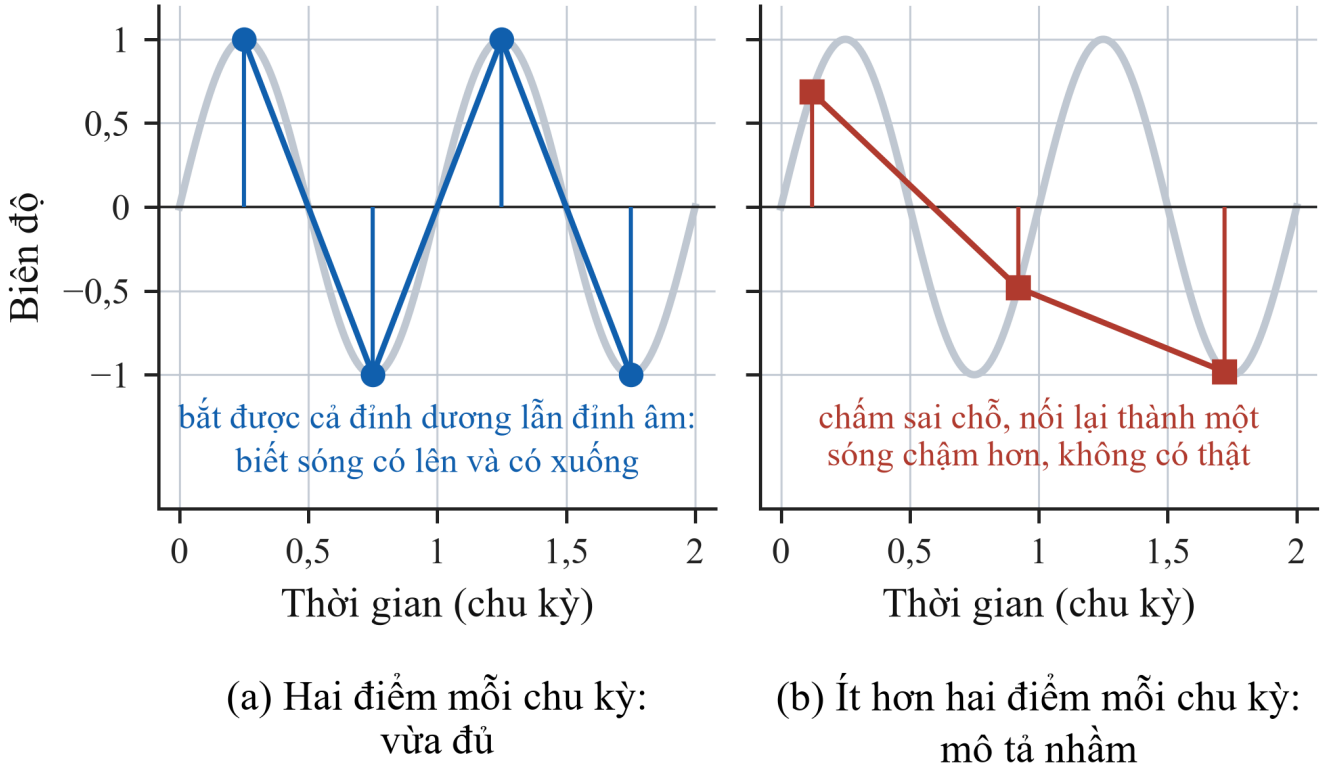
$$f_s > 2f_{\max}\ ; \qquad f_{\text{Nyq}} = \frac{f_s}{2}$$

(3.2)

- Muốn khôi phục trọn vẹn: fs phải lớn hơn hai lần tần số cao nhất fmax của tín hiệu.
- fNyq = fs/2 là tần số cao nhất mà hệ thống số biểu diễn trung thực.
- Trên Nyquist không phải biểu diễn kém đi mà là biểu diễn sai hẳn (mục 3.3).

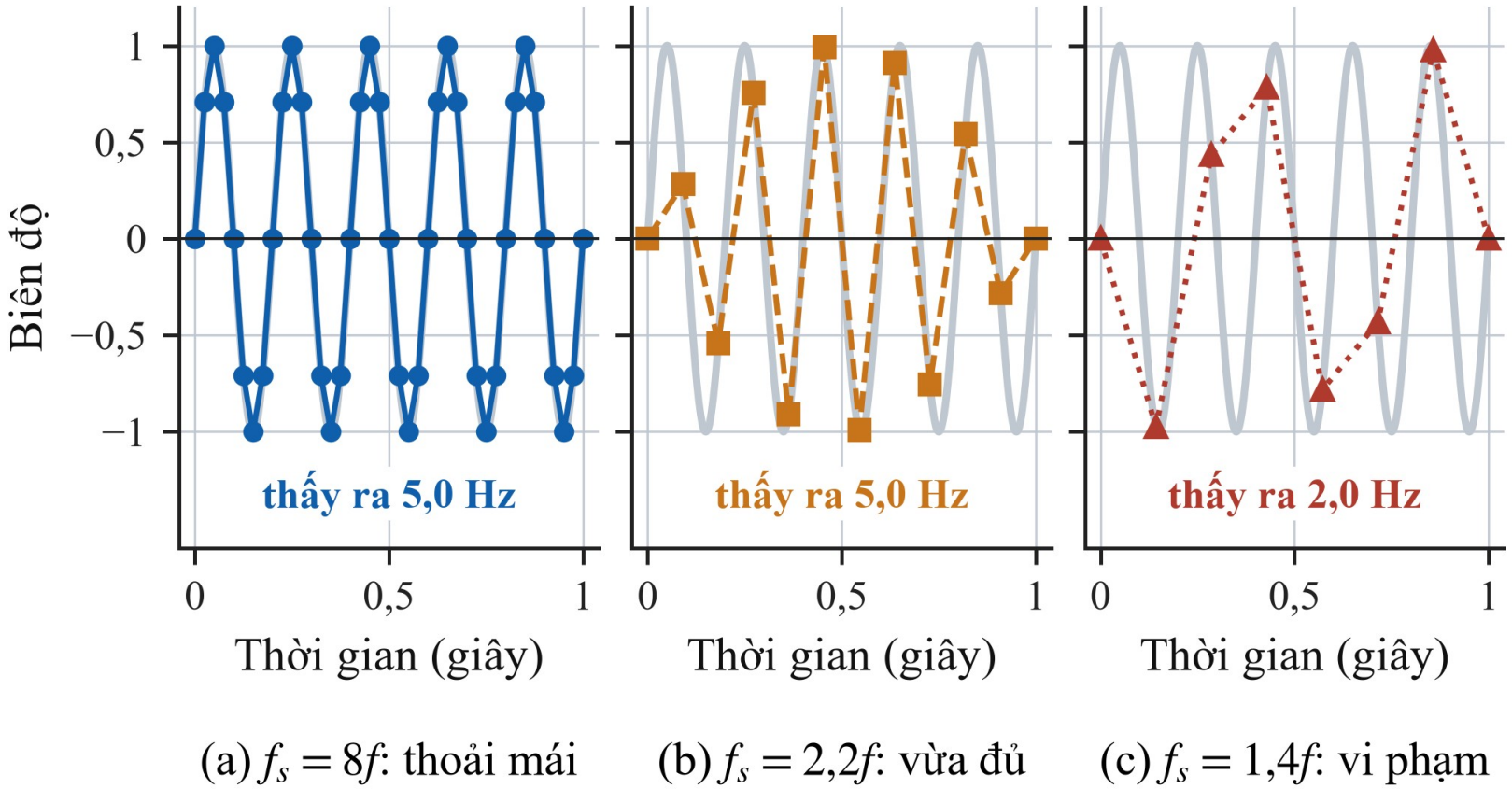
Trực giác: hai điểm đo mỗi chu kỳ

- Sóng có lên và có xuống: một điểm cho phần trên, một cho phần dưới.
- Mỗi chu kỳ cần tối thiểu hai điểm đo.
- Lấy thưa hơn thì các điểm rơi sai chỗ, nối thành một sóng chậm hơn không có thật.



Hình 3.4 — Trực giác hình học của định lý: mỗi chu kỳ cần tối thiểu hai điểm đo

Cùng một sóng 5 Hz, ba mật độ lấy mẫu



Hình 3.5 — Cùng một sóng 5 Hz, ba mật độ lấy mẫu khác nhau

Vì sao là lớn hơn hẳn 2·fmax

Ở đúng 2·fmax, mẫu có thể rơi trúng chỗ tín hiệu cắt không: mọi mẫu bằng 0. Bộ lọc cũng cần khoảng dư để suy giảm dần. CD chọn 44.100 Hz cho dải 20 kHz, dư 4.100 Hz so với 40.000 Hz.

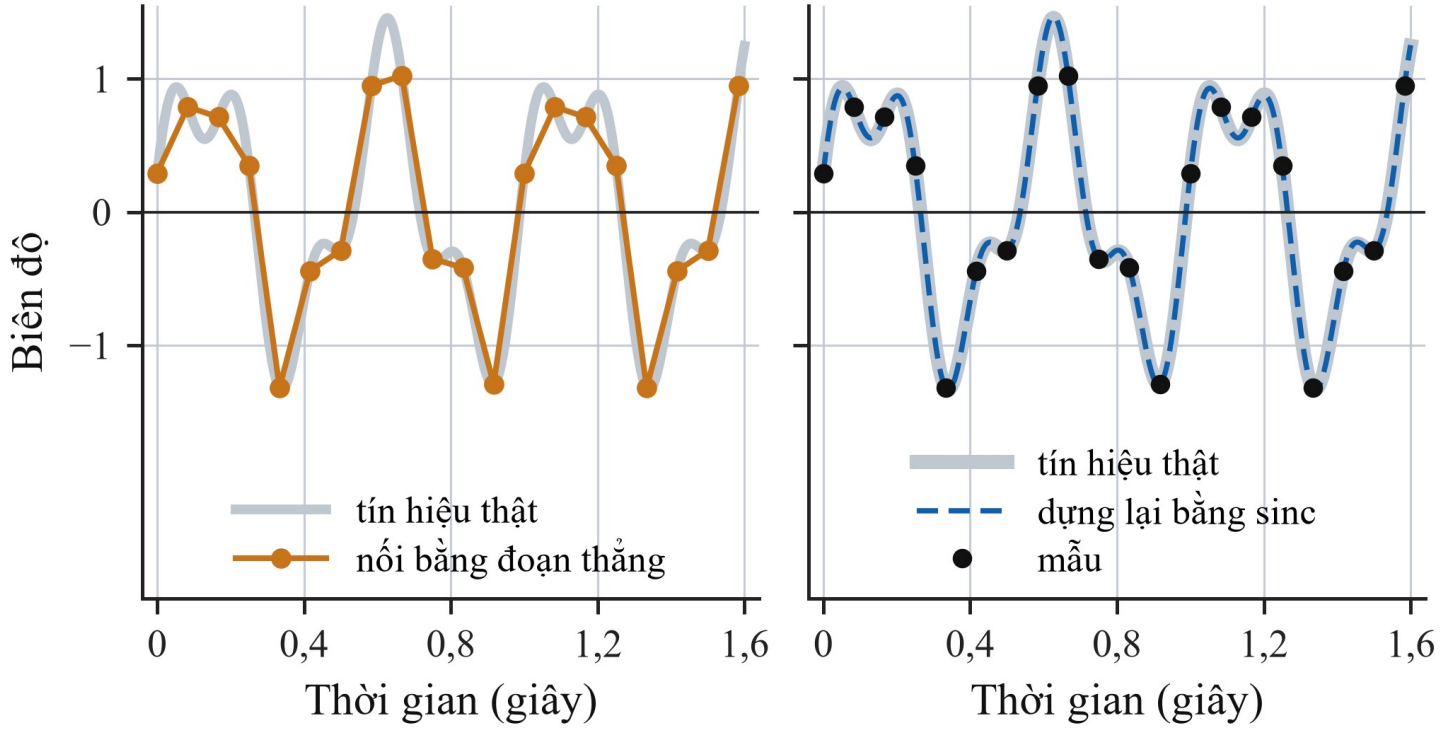
Công thức dựng lại tín hiệu

$$x(t) = \sum_{n=-\infty}^{+\infty} x[n]\,\mathrm{sinc}\left(\frac{t - nT}{T}\right)$$

(3.3)

- Mỗi mẫu x[n] toả ra một hàm sinc; cộng tất cả lại được đúng tín hiệu gốc.
- Không phải nối các chấm bằng đoạn thẳng.
- Phản trực giác nhất của chương: giữa hai mẫu không lưu gì, vậy mà tín hiệu vẫn xác định chính xác.

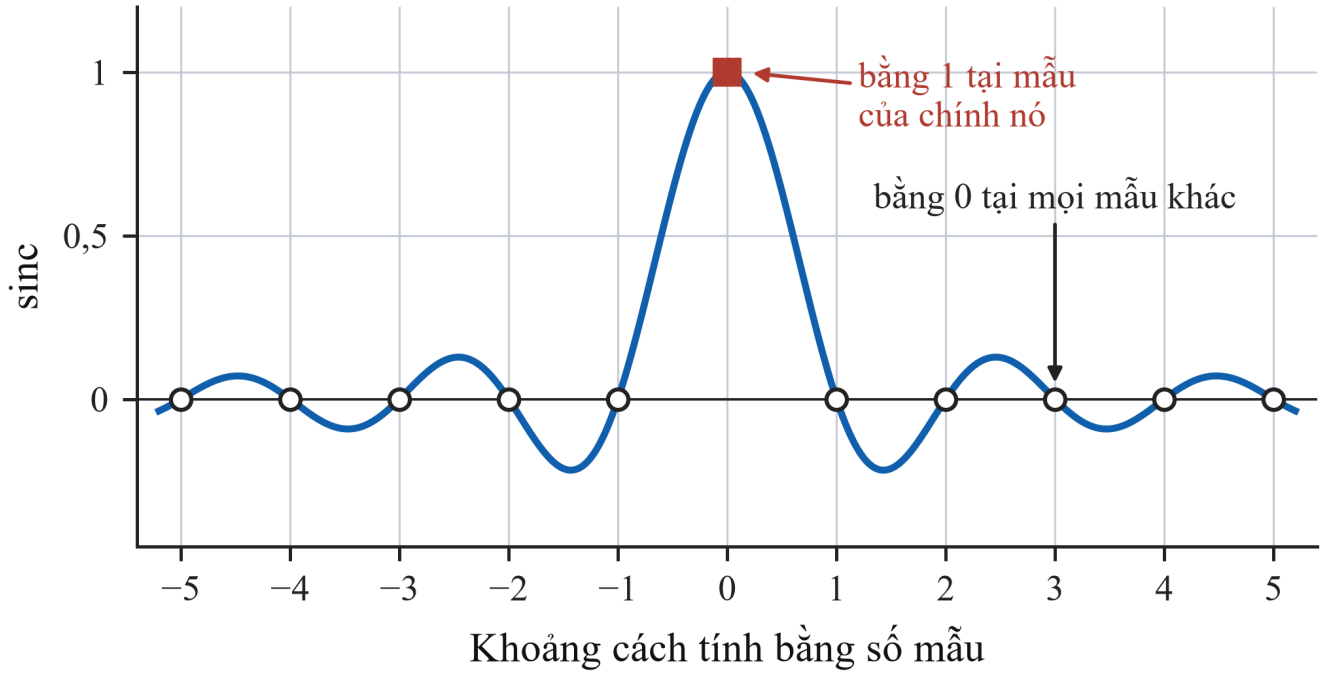
Nối đoạn thẳng thì sai; sinc thì trùng khít



Hình 3.6 — Nối chấm bằng đoạn thẳng thì sai; dựng lại bằng sinc thì trùng khít

Vì sao tổng các hàm sinc đi đúng qua mọi mẫu

- Tại mốc của chính nó: sinc bằng 1; tại mọi mốc khác: bằng đúng 0.
- Nên tại mỗi mốc chỉ một hàm đóng góp, tổng đi đúng qua mẫu.
- Giữa hai mẫu mọi hàm cùng đóng góp: đó là chỗ thông tin được khôi phục.
- Dạng bậc thang chỉ là cách vẽ; bộ chuyển đổi ngược dựng đường cong trơn. Bậc thang là chuyện của lượng tử hoá.



Hình 3.7 — Hàm sinc: bằng 1 tại mẫu của chính nó, bằng 0 tại mọi mẫu khác

Các tần số lấy mẫu chuẩn và lý do chọn

Tần số	Nyquist	Dùng ở đâu	Vì sao chọn
8.000 Hz	4,0 kHz	Điện thoại	đủ cho dải 300–3.400 Hz của tiếng nói
16.000 Hz	8,0 kHz	Nhận dạng tiếng nói	gấp đôi, giữ được phụ âm gió
22.050 Hz	11,0 kHz	Web đời đầu	đúng một nửa chuẩn CD
44.100 Hz	22,1 kHz	CD	vừa khung hình máy ghi video thời đó
48.000 Hz	24,0 kHz	Phim, truyền hình	chia chẵn cho 24, 25, 30 hình/giây
96.000 Hz	48,0 kHz	Phòng thu, gốc 48	gấp đôi chuẩn phim
192.000 Hz	96,0 kHz	Lưu trữ, xử lý chuyên sâu	chỗ cho dịch cao độ và hiệu ứng

Rút gọn từ Bảng 3.2 (7 trong 9 dòng; lược 32.000 Hz và 88.200 Hz).

Tần số lấy mẫu càng cao càng hay?

- Đã vượt ngưỡng Nyquist thì tín hiệu khôi phục trọn vẹn, không có chuyện gần hơn nữa.
- 96 kHz vẫn có ích, nhưng ở khâu xử lý: bộ lọc dốc thoải hơn, ít méo pha.
- Dịch cao độ, kéo giãn thời gian, mô phỏng phi tuyến sinh thành phần cao và cần chỗ chứa.
- Cái giá: dung lượng và tải CPU gấp đôi.

Không được gì ở khâu nghe

Tai đã không nghe được trên 20 kHz. Ghi thêm phần trên đó không làm phần dưới nghe khác đi chút nào. Đây là hiểu lầm phổ biến nhất về số hoá.

Được khá nhiều ở khâu xử lý

Bộ lọc chống chồng phổ có chỗ dốc thoải hơn, ít méo pha hơn. Dịch cao độ, kéo giãn thời gian, mô phỏng phi tuyến đều cần dư dải.

Thu và xử lý ở 96 kHz thì có lý; phát hành ở 96 kHz thì gần như vô nghĩa.

Cái giá thực tế: dung lượng gấp đôi, tải CPU gấp đôi, độ trễ khối đệm không giảm.

Nguyên tắc chung: chọn tham số theo công việc, không theo con số lớn nhất có thể.

Hình 3.9 — Tăng tần số lấy mẫu được gì và không được gì

Thu và xử lý ở 96 kHz thì có lý. Phát hành ở 96 kHz thì gần như vô nghĩa.

Nguyên tắc của cả học phần: chọn tham số theo công việc, không theo con số lớn nhất có thể.

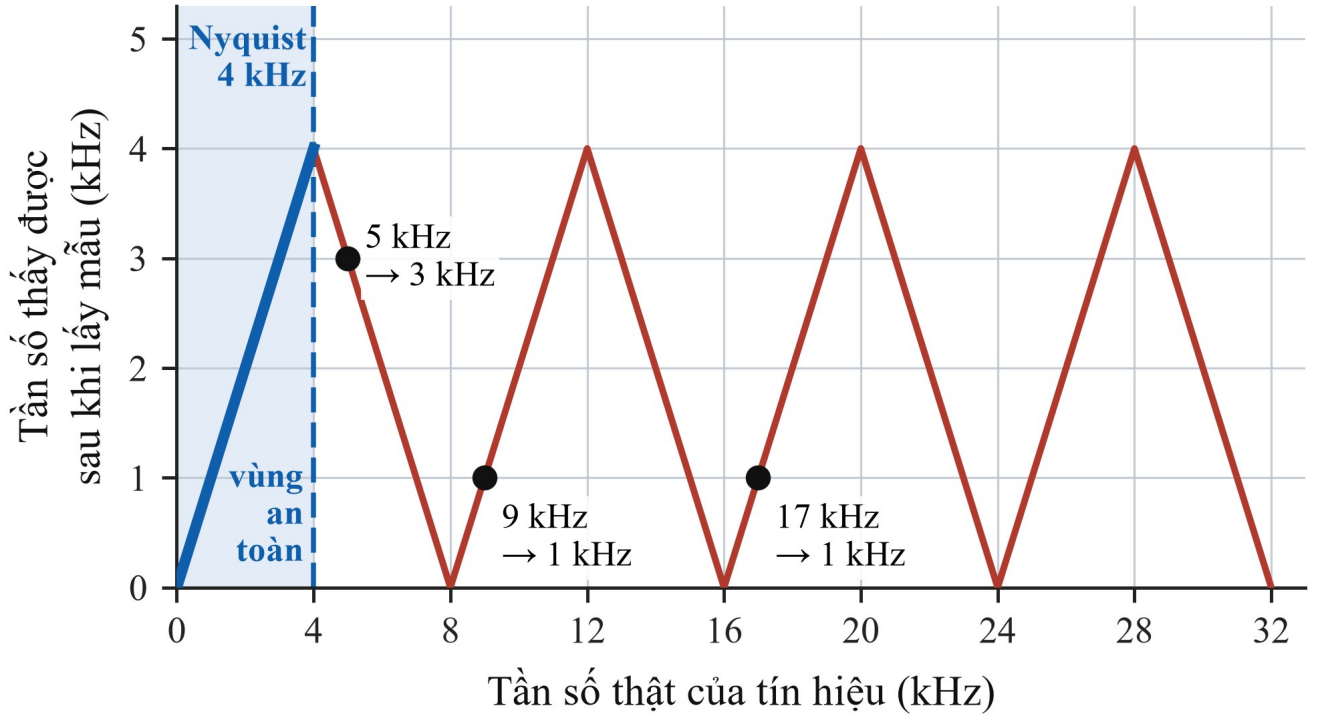
MỤC 3.3

Chồng phổ và bộ lọc chặn nó

- Tần số vượt Nyquist không biến mất: nó gập xuống thành tần số thấp giả.
- Không sửa được sau khi lấy mẫu; phải chặn từ trước.
- Chồng phổ cũng xảy ra trong mã chương trình, không chỉ ở phần cứng.

Bản đồ gập tần số

- Mọi tần số trên Nyquist rơi về một tần số trong vùng an toàn.
- Đường gấp khúc giống tờ giấy gấp nhiều lần: nguồn gốc tên gọi gập.
- Phép gập không thuận nghịch: không biết 3.000 Hz vốn là 3.000 hay 5.000 Hz.



Hình 3.10 — Bản đồ gập: mọi tần số trên Nyquist đều rơi về một tần số trong vùng an toàn

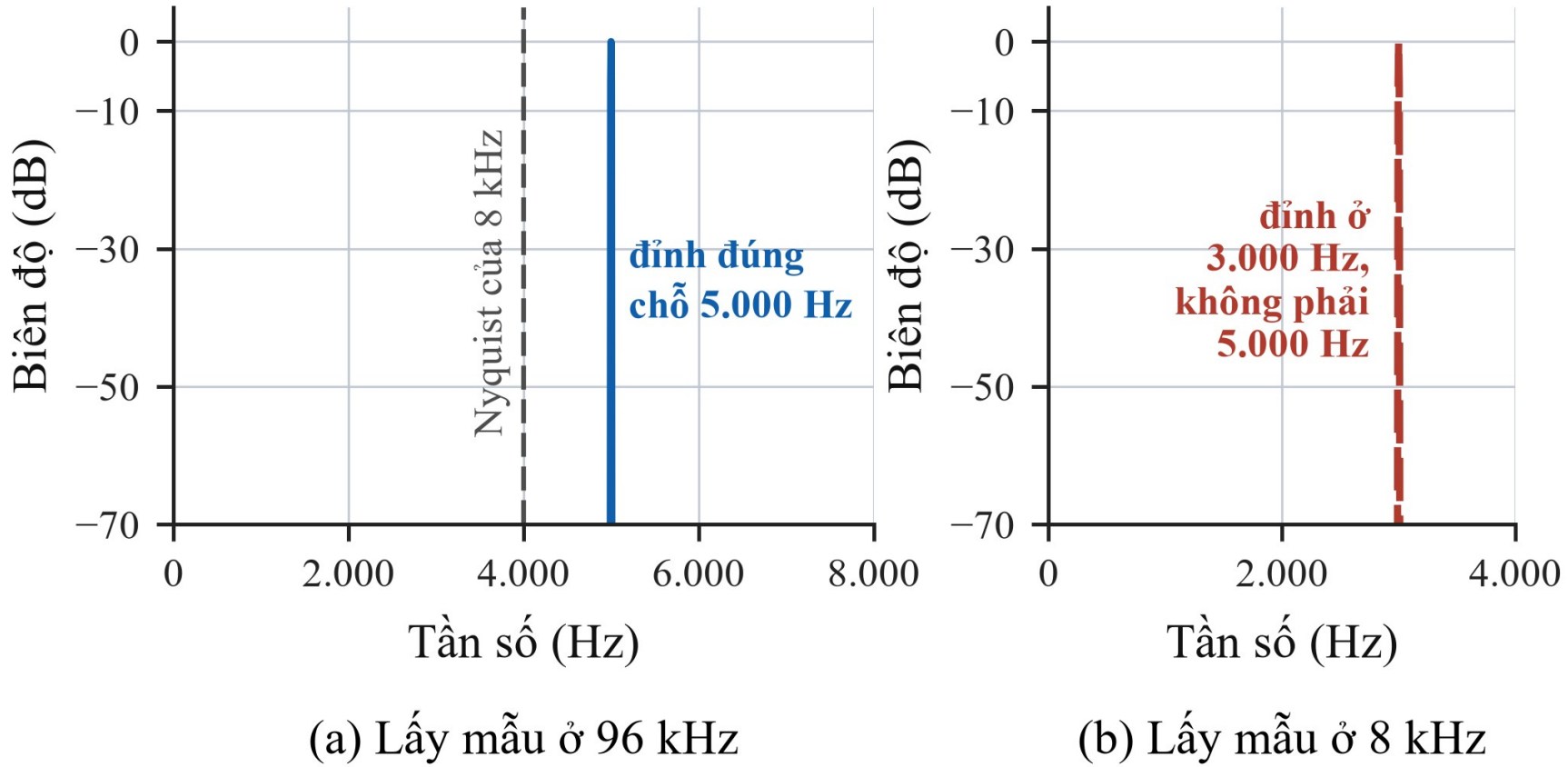
Tần số thấy được

$$f_{\text{thay}} = |f - k f_s|, \qquad k = \text{round}\left(\frac{f}{f_s}\right)$$

(3.4)

- f là tần số thật; k là số nguyên gần nhất của f/fs.
- fthay là tần số mà bộ phân tích đo ra.
- Ví dụ: 5.000 Hz lấy mẫu ở 8.000 Hz cho k = 1, fthay = 3.000 Hz.

Cùng sóng 5.000 Hz, hai tần số lấy mẫu, hai kết quả



Hình 3.11 — Kiểm thật trên tín hiệu: cùng một sóng 5.000 Hz, hai tần số lấy mẫu, hai kết quả

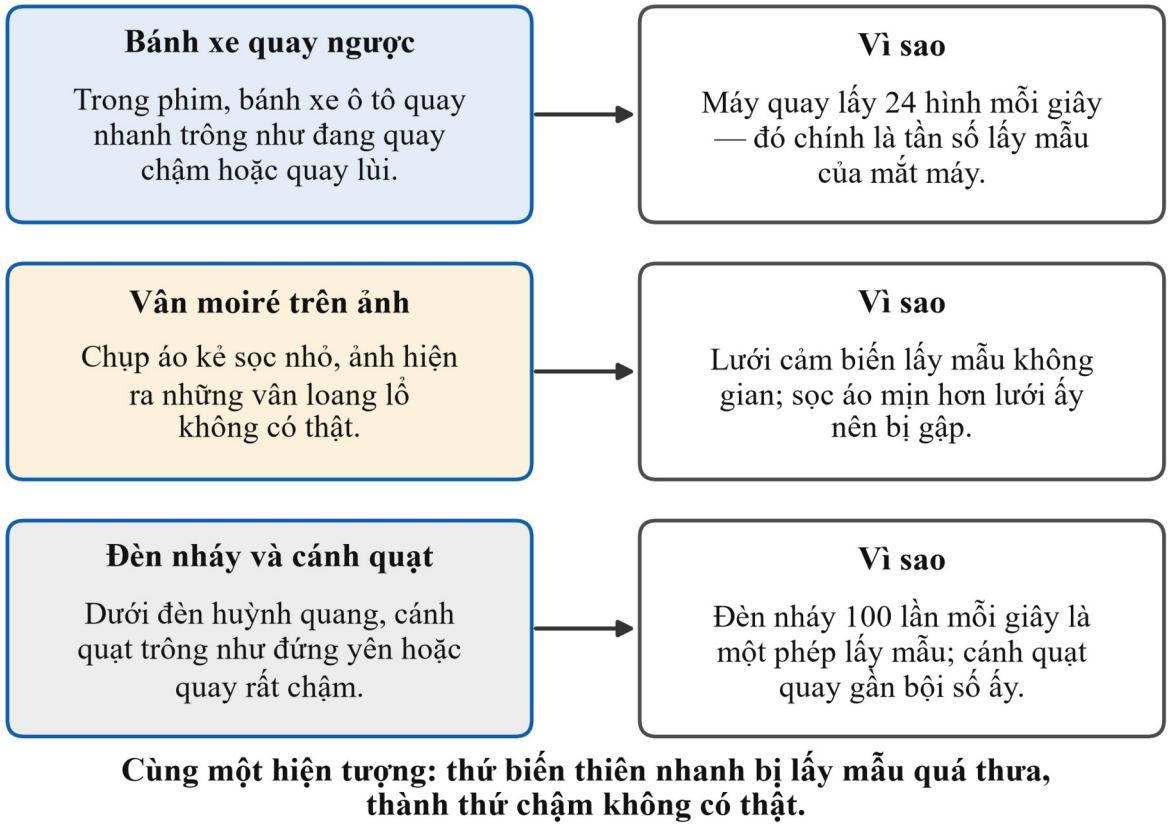
Ví dụ gập ở tần số lấy mẫu 8.000 Hz

Tần số thật	Vượt Nyquist 4.000 Hz?	Tần số thấy được
1.000 Hz	không	1.000 Hz
3.000 Hz	không	3.000 Hz
5.000 Hz	có	3.000 Hz (giả)
7.000 Hz	có	1.000 Hz (giả)
9.000 Hz	có	1.000 Hz (giả)
17.000 Hz	có	1.000 Hz (giả)

1.000, 9.000 và 17.000 Hz cho dãy số giống hệt nhau: không phải khó phân biệt mà là không có gì để phân biệt.

Cùng một hiện tượng, khác đối tượng bị lấy mẫu

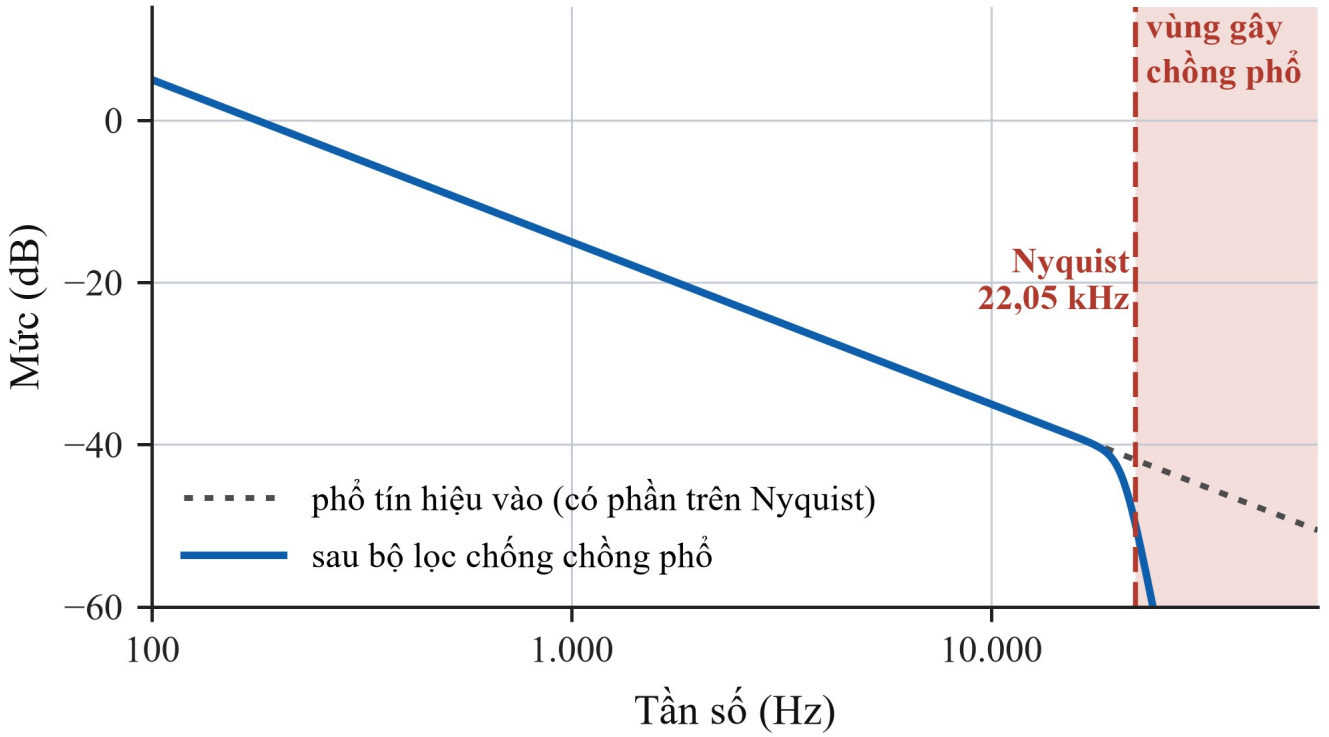
- Bánh xe quay ngược, vân moiré, cánh quạt dưới đèn nháy.
- Máy quay 24 hình/giây là một phép lấy mẫu.
- Bánh xe quay nhanh hơn Nyquist của nó thì bị mô tả thành quay chậm hoặc quay lùi.



Hình 3.12 — Bánh xe quay ngược, vân moiré, cánh quạt dưới đèn nháy: cùng một hiện tượng

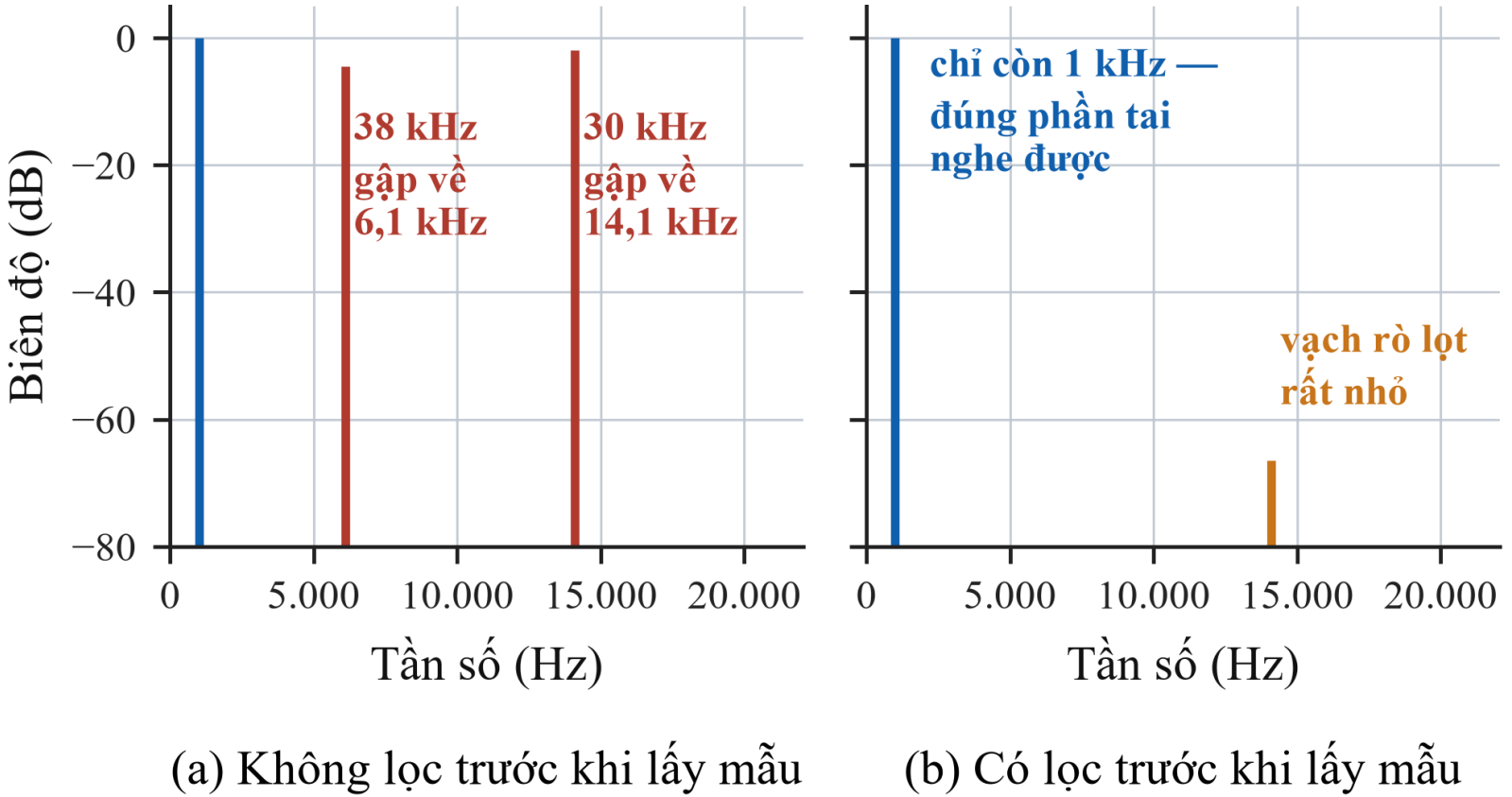
Chặn trước khi lấy mẫu

- Không có cách sửa chồng phổ sau khi đã lấy mẫu.
- Giải pháp duy nhất: bộ lọc thông thấp ngay trước bộ chuyển đổi, khi tín hiệu còn analog.
- Đây là khối lọc chống răng cưa trong sơ đồ giao diện âm thanh ở chương 2.



Hình 3.13 — Bộ lọc chống chồng phổ cắt phần trên Nyquist trước khi lấy mẫu

Tín hiệu có thành phần siêu âm: lấy thẳng và lấy sau khi lọc



Hình 3.14 — Cùng một tín hiệu có thành phần siêu âm: lấy mẫu thẳng và lấy mẫu sau khi lọc

Chồng phổ biến thứ vô hại thành thứ có hại

Siêu âm 30 kHz và 38 kHz tai không nghe được. Lấy mẫu ở 44,1 kHz không lọc, chúng thành 14,1 kHz và 6,1 kHz giữa dải nghe. Bộ lọc chống chồng phổ là khối bắt buộc, không phải tuỳ chọn.

Ba nơi mã tự tạo ra chồng phổ, và hai cách xử lý

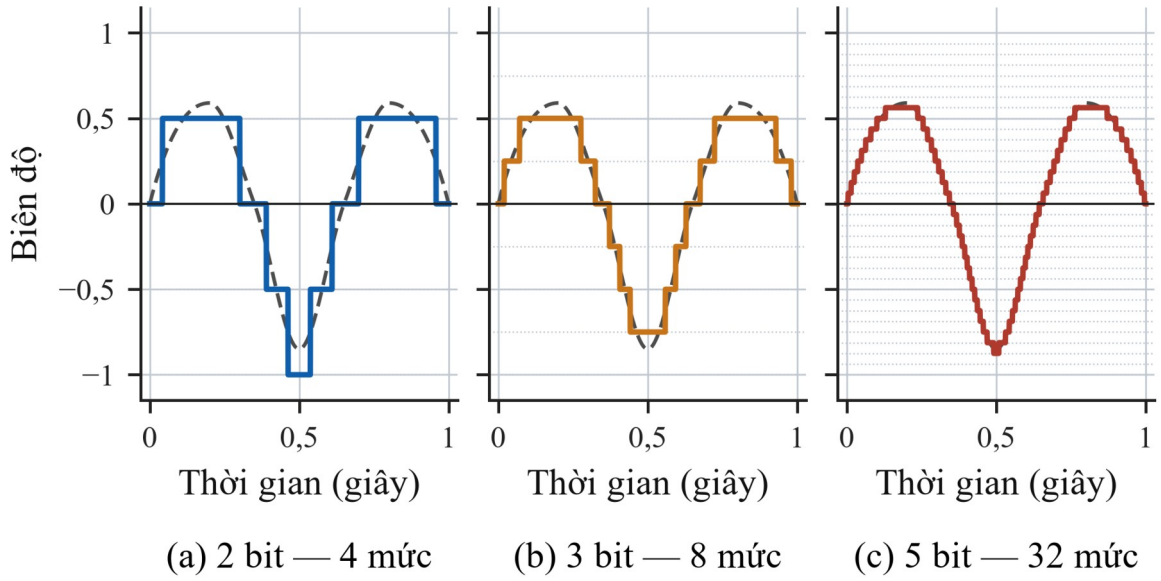
- **Sinh tín hiệu bằng công thức:** hài vượt Nyquist gập xuống; dừng cộng ở hài cuối còn dưới Nyquist.
- **Hạ tần số lấy mẫu:** bỏ mẫu mà không lọc trước thì mọi thứ trên Nyquist mới gập xuống.
- **Làm méo phi tuyến:** cắt ngọn, bão hoà sinh hài mới ở tần số cao rồi gập xuống (gặp lại ở chương 8).
- **Cách xử lý 1:** lọc thông thấp ở Nyquist mới trước mỗi phép hạ tần số.
- **Cách xử lý 2:** oversampling: nâng fs, làm phép phi tuyến, lọc, rồi hạ về.

Nguyên tắc chung: giữ mọi thành phần dưới Nyquist tại mọi bước, không chỉ ở bước đầu.

MỤC 3.4

Lượng tử hoá, độ sâu bit và dither

- Sau lấy mẫu, mỗi giá trị vẫn là số thực vô hạn chữ số.
- Lượng tử hoá: làm tròn mỗi mẫu về một trong số hữu hạn mức.
- Số mức do độ sâu bit quyết định.



(a) 2 bit — 4 mức (b) 3 bit — 8 mức (c) 5 bit — 32 mức

Độ sâu bit quyết định số mức

$$M = 2^b ; \qquad \Delta = \frac{2A_{\max}}{2^b}$$

(3.5)

- b là độ sâu bit; M = 2^b là số mức.
- Δ là bước lượng tử; Amax là biên độ tối đa.
- Thêm một bit thì số mức nhân đôi, bước lượng tử giảm một nửa.

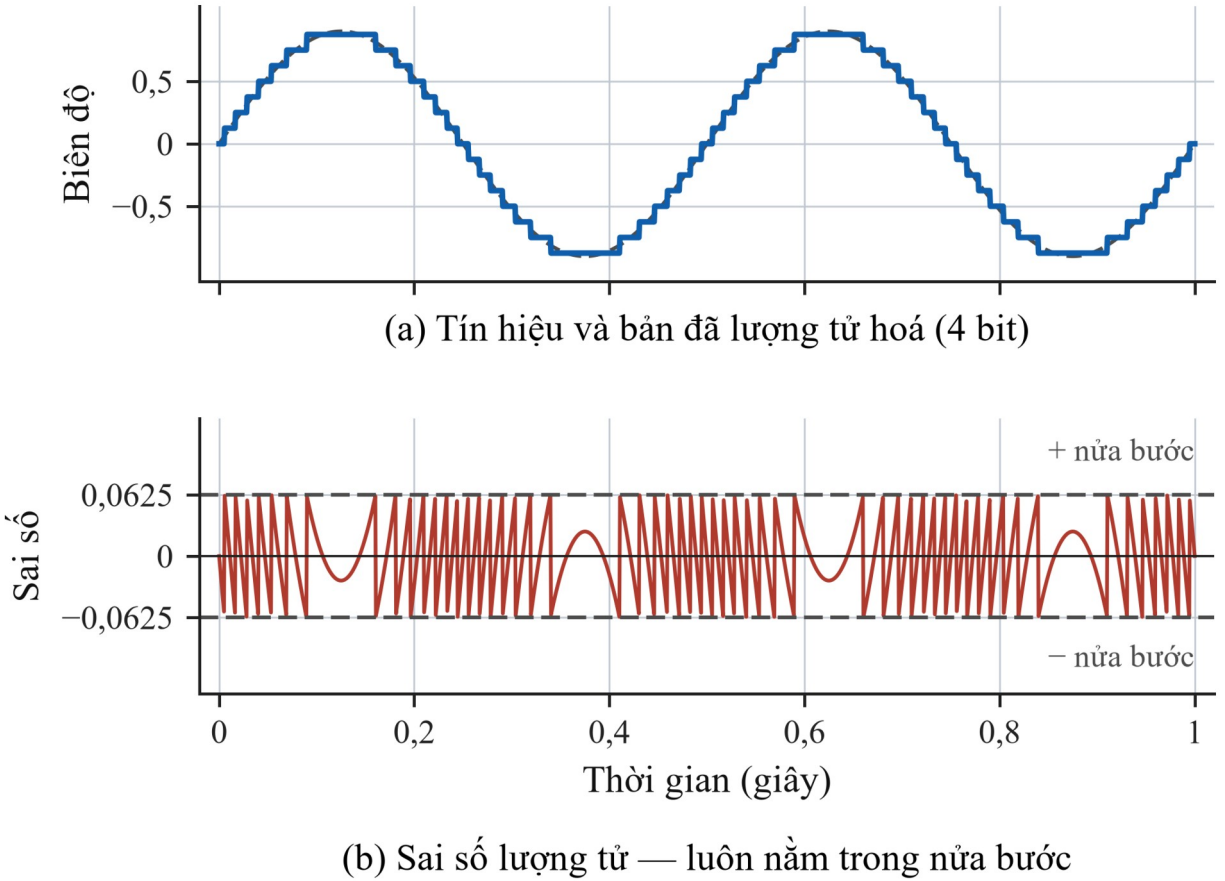
Độ sâu bit, số mức và dải động

Độ sâu bit	Số mức	Bước lượng tử	Dải động
8 bit	256	0,781 %	49,9 dB
12 bit	4.096	0,049 %	74,0 dB
16 bit	65.536	30,52 phần triệu	98,1 dB
20 bit	1.048.576	1,91 phần triệu	122,2 dB
24 bit	16.777.216	0,12 phần triệu	146,2 dB

Bước lượng tử tính theo phần của toàn thang.

Bị chặn trong nửa bước, nhưng có quy luật

- Nhiễu lượng tử là chênh lệch giữa giá trị thật và giá trị đã làm tròn.
- Bị chặn cứng trong nửa bước nên năng lượng rất nhỏ.
- Nhưng năng lượng chưa phản ánh đầy đủ ảnh hưởng của nó: xem mục 3.4.3.



Hình 3.16 — Sai số lượng tử nhìn riêng ra: bị chặn trong nửa bước, nhưng có quy luật

Mỗi bit thêm 6,02 dB

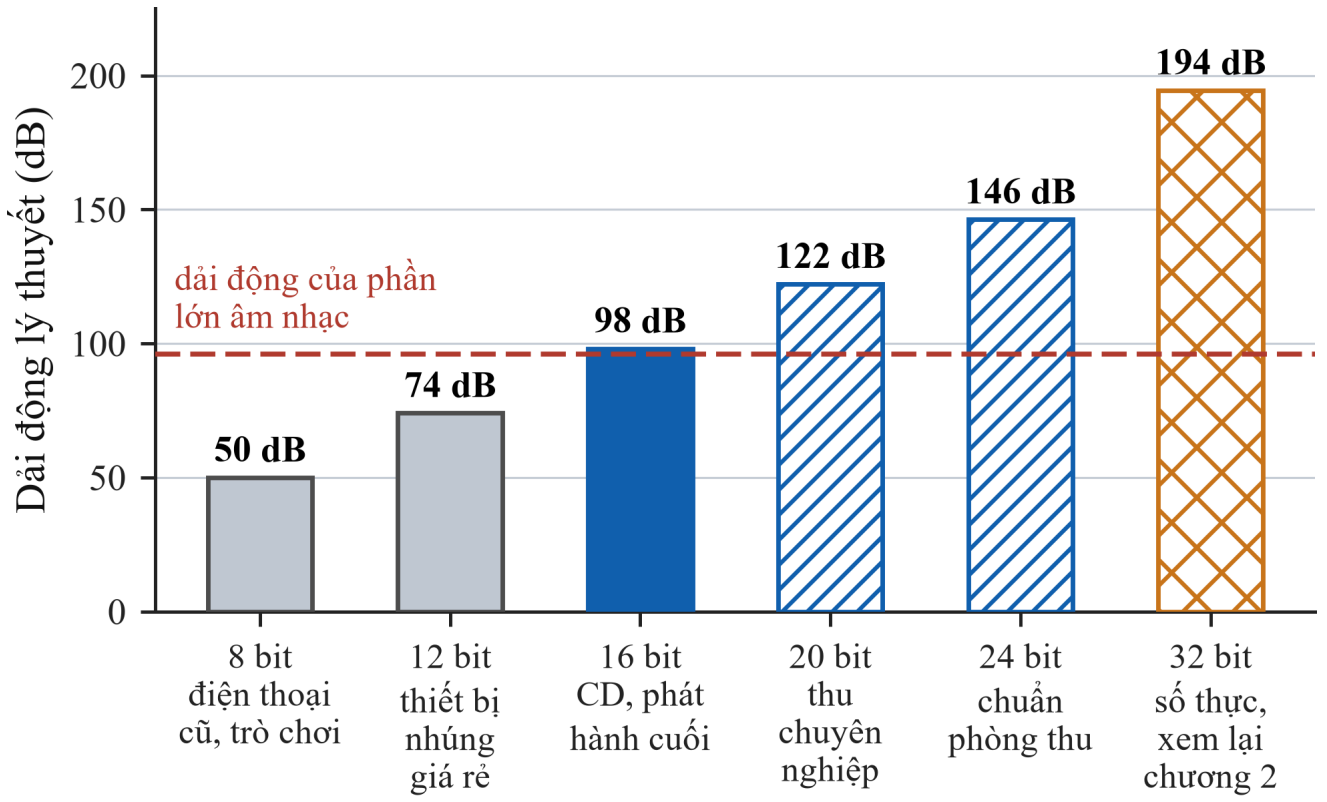
$$e_{\text{rms}} = \frac{\Delta}{\sqrt{12}}; \qquad \text{SQNR} \approx 6{,}02\,b + 1{,}76 \quad [\text{dB}]$$

(3.6)

- e_rms = Δ/√12 là giá trị hiệu dụng của sai số phân bố đều trong một bước.
- SQNR (Signal-to-Quantization-Noise Ratio – tỉ số tín hiệu trên nhiễu lượng tử) là tỉ số tín hiệu trên nhiễu lượng tử; mỗi bit cho thêm 6,02 dB.
- Hằng số 1,76 dB đến từ √12.

Thu ở 24 bit, phát hành 16 bit

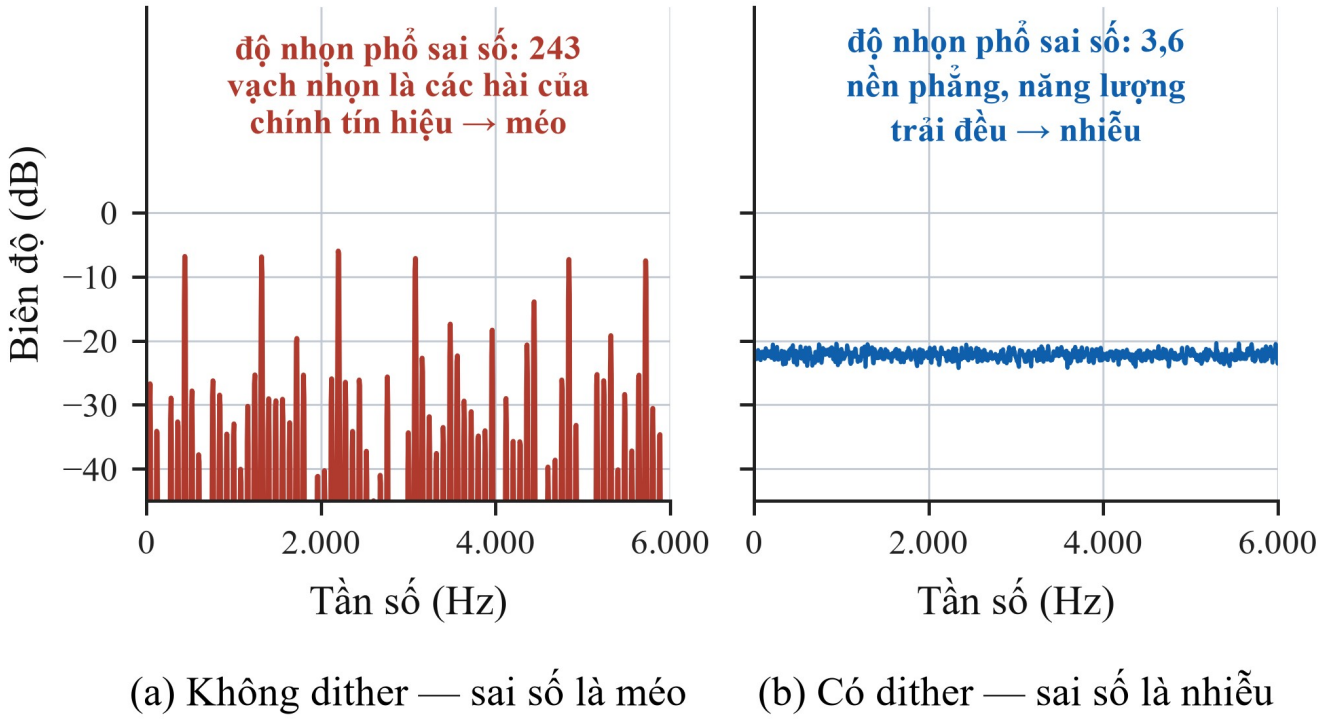
- Không phải vì tai nghe được 146 dB.
- Lúc thu chưa biết đỉnh ở đâu nên để headroom rộng, tín hiệu nằm thấp.
- Mọi phép xử lý trung gian đều làm tròn một lần; bắt đầu từ 24 bit thì sau mười phép vẫn còn dư.
- Đây là gain staging của chương 2 trong ngôn ngữ số.



Hình 3.17 — Dải động theo độ sâu bit, và vùng dải động của phần lớn âm nhạc

Đổi méo lấy nhiễu

- Sai số có quy luật: cùng giá trị vào thì cùng sai số ra, phổ có vạch nhọn, tai nghe thành méo.
- Vấn đề là hình dạng của sai số, không phải mức năng lượng.
- Dither: cộng nhiễu ngẫu nhiên rất nhỏ trước khi làm tròn, biến sai số thành nhiễu nền phẳng.



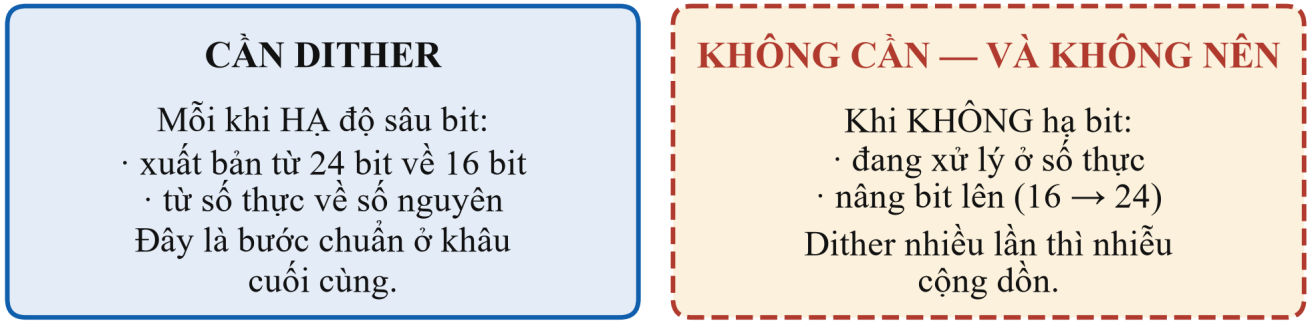
Hình 3.18 — Phổ của riêng sai số: không dither thì có vạch nhọn, có dither thì phẳng

Độ nhọn phổ giảm từ 243 xuống 3,6

Năng lượng sai số tăng vài decibel vì đã cộng thêm nhiễu. Đánh đổi có lợi: tai chịu nhiễu nền đều dễ hơn méo bám theo tín hiệu. Dạng chuẩn là TPDF (Triangular PDF – nhiễu phân bố tam giác): phân bố tam giác, biên độ một bước lượng tử.

Đúng một lần, ở bước hạ bit cuối cùng

- Giống quy tắc gain staging ở chương 2 và nén có tổn hao ở mục 3.6.
- Mọi phép biến đổi không hoàn tác được nên dồn về đúng một lần, ở bước cuối.
- Đây là nguyên tắc chung của lĩnh vực, không riêng gì dither.



Quy tắc một câu: dither ĐÚNG MỘT LẦN, ở đúng bước hạ bit cuối cùng trước khi phát hành.

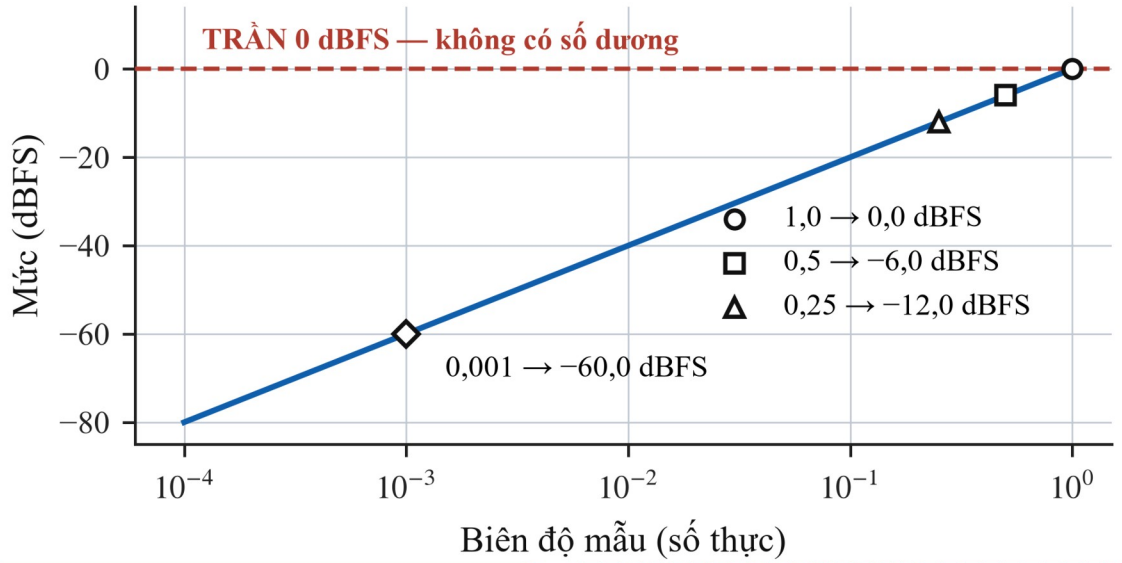
Lỗi thường gặp của người mới: bật dither ở mọi khâu xuất trung gian, làm nền nhiễu dâng lên vô ích qua từng bước.

Hình 3.19 — Dither đúng một lần, ở đúng bước hạ bit cuối cùng

MỤC 3.5

dBFS và cách máy tính lưu mẫu

- Mục ngắn nhất nhưng dùng nhiều nhất: mọi chương sau dựa trên quy ước ở đây.
- Thang dB thứ ba: dBFS, mốc là toàn thang của bộ chuyển đổi.
- Hai cách lưu mẫu: số nguyên hoặc số thực.



BA THANG DECIBEL ĐÃ HỌC

dB SPL: mốc là áp suất 20 µPa (chương 1)
dBu, dBV: mốc là điện áp (chương 2)
dBFS: mốc là toàn thang của bộ chuyển đổi

QUY TẮC BẤT DI BẤT DỊCH

Chỉ được trừ hai con số CÙNG một thang. Không có phép đổi trực tiếp giữa ba thang — muốn đổi phải biết cả chuỗi tín hiệu ở chương 2.

Đổi giữa biên độ mẫu và dBFS

$$L_{\mathrm{FS}} = 20 \log_{10} |a| \iff a = 10^{L_{\mathrm{FS}}/20}$$

(3.7)

- a là biên độ mẫu, với trần là 1,0; L_FS là mức theo dBFS.
- dB SPL (Sound Pressure Level – mức áp suất âm) và dBu đo đại lượng vật lý; dBFS đo so với trần nên mọi tín hiệu hợp lệ đều âm.
- Thấy dBFS dương là biết đã tràn. Chỉ được trừ hai con số cùng một thang.

Vài mốc dBFS nên thuộc

Biên độ mẫu	Mức dBFS	Ý nghĩa
1,0	0,00 dBFS	Trần; chạm mức này là sát ngưỡng méo cắt
0,5	−6,02 dBFS	Nửa toàn thang; chia đôi biên độ là −6 dB
0,126	−18,00 dBFS	Mức làm việc chuẩn của phòng thu
0,0010	−60,00 dBFS	Đoạn rất nhỏ; còn xa sàn nhiễu của 16 bit

Đổi số thực sang số nguyên b bit

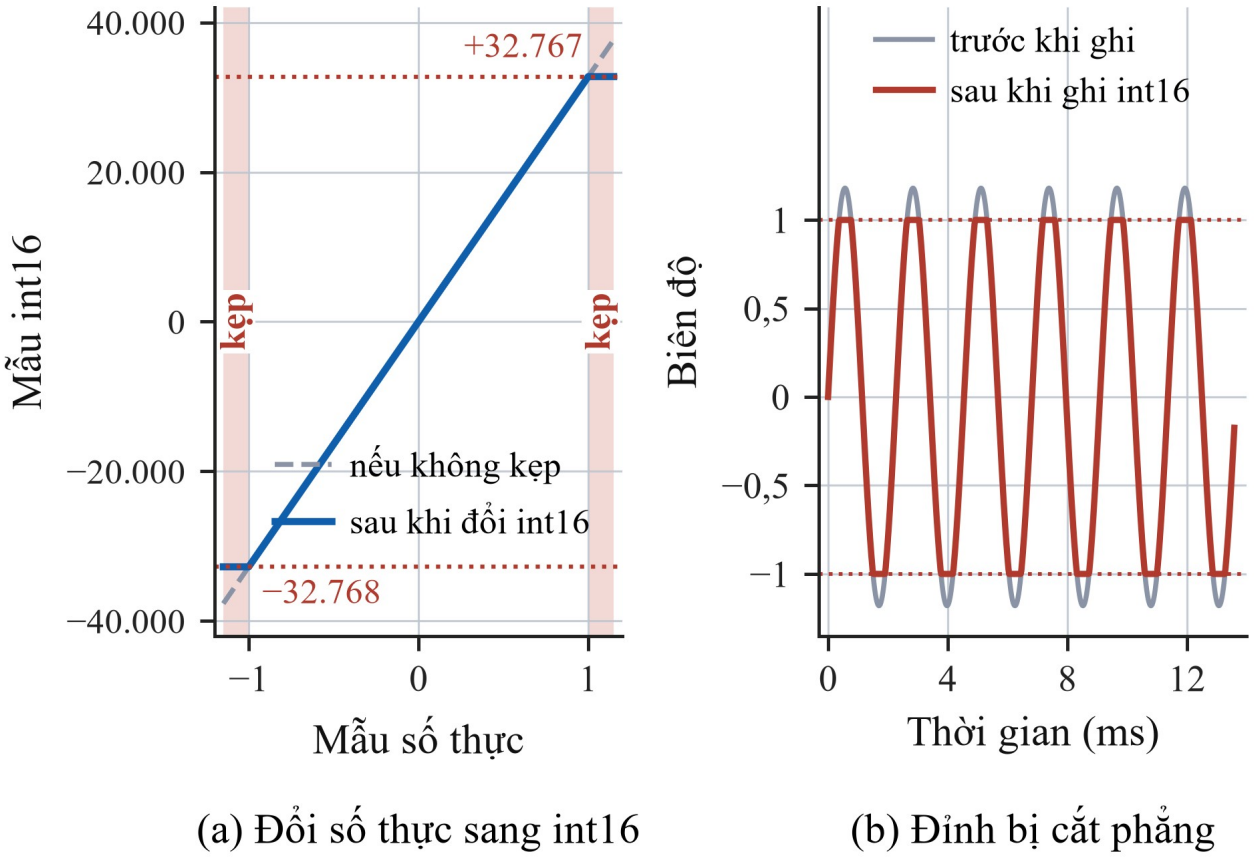
$$i = \text{round}(a \cdot 2^{b-1}), \qquad -2^{b-1} \le i \le 2^{b-1} - 1$$

(3.8)

- a là mẫu số thực trong [−1; +1]; i là số nguyên tương ứng.
- Dải int16: từ −32.768 tới 32.767, không đối xứng.
- Mẫu +1,0 bị kẹp về 32.767 chứ không thành 32.768.

Dải int16 và hiện tượng kẹp biên

- Chênh một đơn vị là vô hại về mặt nghe.
- Nhưng cho thấy phép đổi float sang int có chi tiết cần đọc tài liệu thư viện.
- Không tự viết mà không đối chiếu.



Hình 3.21 — Đổi số thực sang int16: dải không đối xứng và hiện tượng kẹp biên

Nguyên tắc cốt lõi khi chọn giữa số nguyên và số thực

SỐ NGUYÊN — int16, int24

Mẫu là số nguyên trong dải của kiểu. int16 chạy từ −32.768 tới +32.767. Gọn, đúng chuẩn tệp WAV. Vượt dải là bị KẸP — mất vĩnh viễn.

SỐ THỰC — float32, float64

Mẫu là số thực, quy ước trong [−1, +1]. Vượt tạm ±1 khi tính trung gian vẫn đúng. Cộng và nhân không tích luỹ sai số. Đây là dạng dùng khi XỬ LÝ.

QUY TẮC VÀNG

Đọc vào → đổi sang SỐ THỰC → xử lý toàn bộ ở số thực → chỉ hạ về số nguyên ở bước GHI CUỐI CÙNG

Làm ngược lại — hạ về số nguyên giữa chừng — thì mỗi bước lại làm tròn một lần, và sai số cộng dồn qua từng khâu.

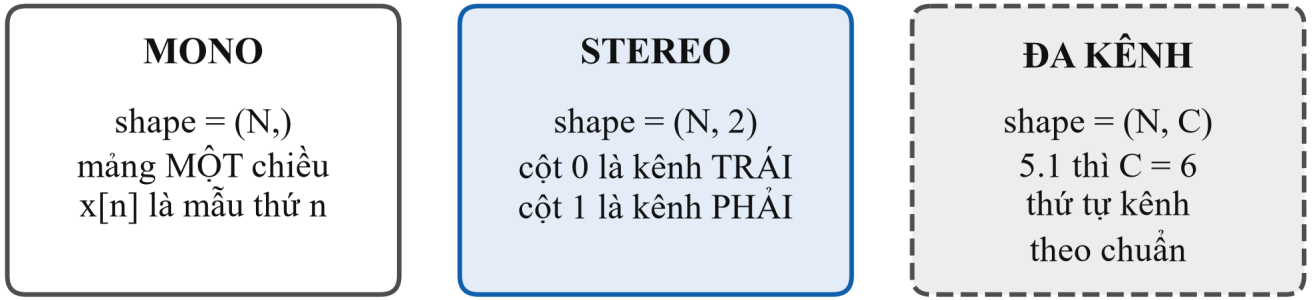
Hình 3.22 — Hai cách lưu mẫu, và nguyên tắc cốt lõi khi chọn giữa chúng

Đọc vào → đổi ngay sang số thực → xử lý ở số thực → chỉ hạ về số nguyên ở bước ghi cuối

Số thực cho phép vượt tạm ±1,0 khi tính trung gian mà kết quả vẫn đúng. Mỗi lần hạ về số nguyên là một lần làm tròn nên hạ giữa chừng thì sai số cộng dồn.

Hình dạng mảng

- Quy ước quyết định cách viết vòng lặp và gọi hàm ở các chương sau.
- Đọc bằng always_2d=True thì mảng luôn hai chiều, x.shape[1] luôn dùng được.
- Ghép về một kênh bằng trung bình cột; hai kênh lệch pha có thể triệt tiêu nhau.
- Ghép xong mất trầm là hiện tượng vật lý, không phải mã sai.



Cách tránh viết hai nhánh mã: đọc bằng always_2d=True thì mảng LUÔN hai chiều, kể cả với tệp mono.

Ghép về một kênh để đo: x.mean(axis=1) — nhưng nhớ rằng ghép hai kênh lệch pha có thể triệt tiêu nhau (chương 2).

Hình 3.23 — Hình dạng mảng của mono, stereo và đa kênh

Bốn quy ước dùng ở mọi chương sau

Câu hỏi	Quy ước của học phần	Vì sao
Mẫu nằm trong khoảng nào?	Số thực trong [−1,0; +1,0]	Vượt tạm khi tính trung gian vẫn đúng; chỉ kẹp ở bước ghi cuối
Đo mức bằng thang nào?	dBFS, mốc 1,0, luôn âm	Thấy số dương là biết đã tràn
Mảng có hình dạng nào?	(N,) hoặc (N, số kênh)	always_2d=True: không cần hai nhánh mã
Hạ về số nguyên khi nào?	Chỉ ở bước ghi tệp cuối	Mỗi lần hạ là một lần làm tròn

MỤC 3.6

Dung lượng, định dạng và nén có tổn hao

- Dung lượng không nén chỉ là một phép nhân.
- Ba nhóm định dạng phân biệt bằng một câu hỏi.
- Nén có tổn hao dựa trên ba điểm mù của tai, và không hoàn tác được.



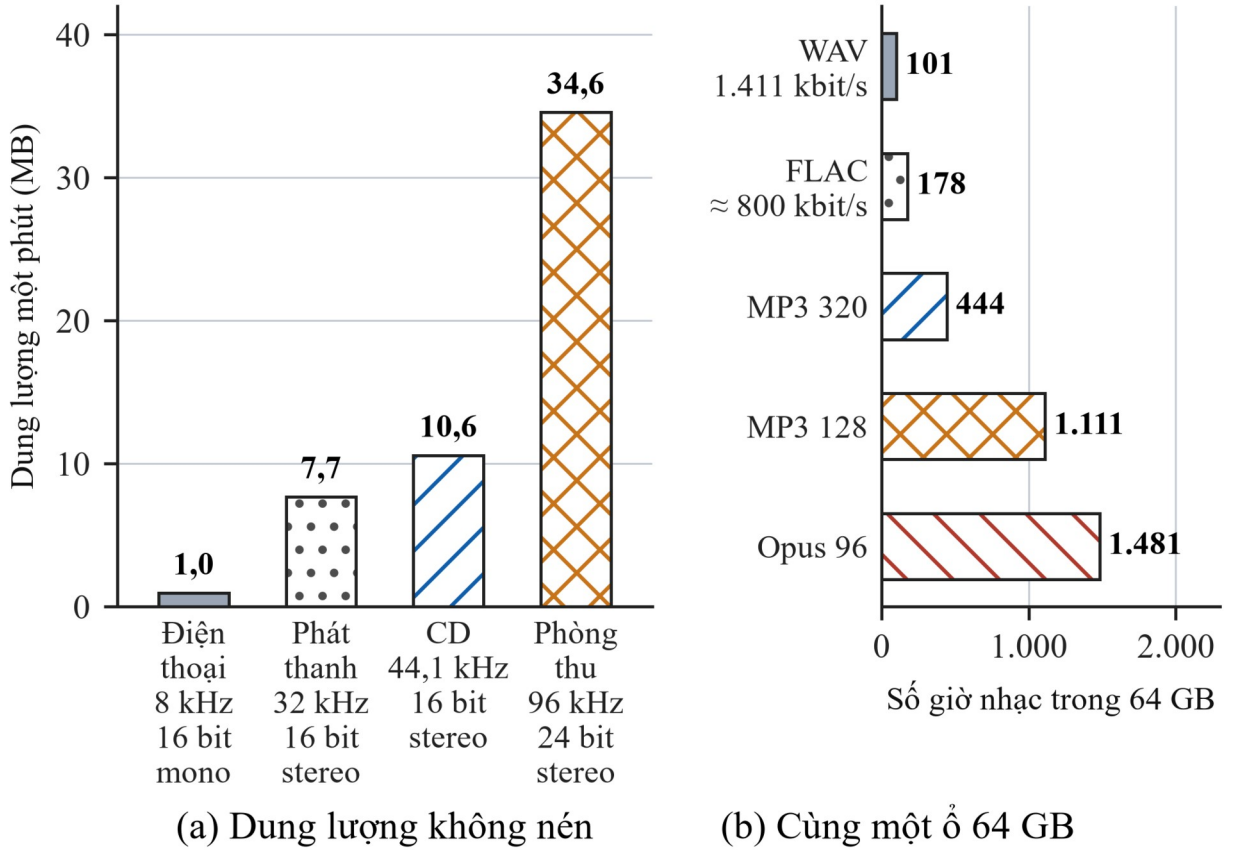
Dung lượng tệp WAV

$$S = f_s \cdot \frac{b}{8} \cdot N_{\text{ch}} \cdot t \quad [\text{byte}]$$

(3.9)

- fs là tần số lấy mẫu, b độ sâu bit, Nch số kênh, t thời lượng (giây).
- Chia 8 để đổi bit thành byte.
- Một phút chất lượng CD: 10,58 MB.

Dung lượng một phút ở các cấu hình, và số giờ trong ổ 64 GB



Hình 3.24 — Dung lượng một phút ở các cấu hình khác nhau, và số giờ chứa được trong một ổ 64 GB

Tốc độ bit và bit cho mỗi mẫu

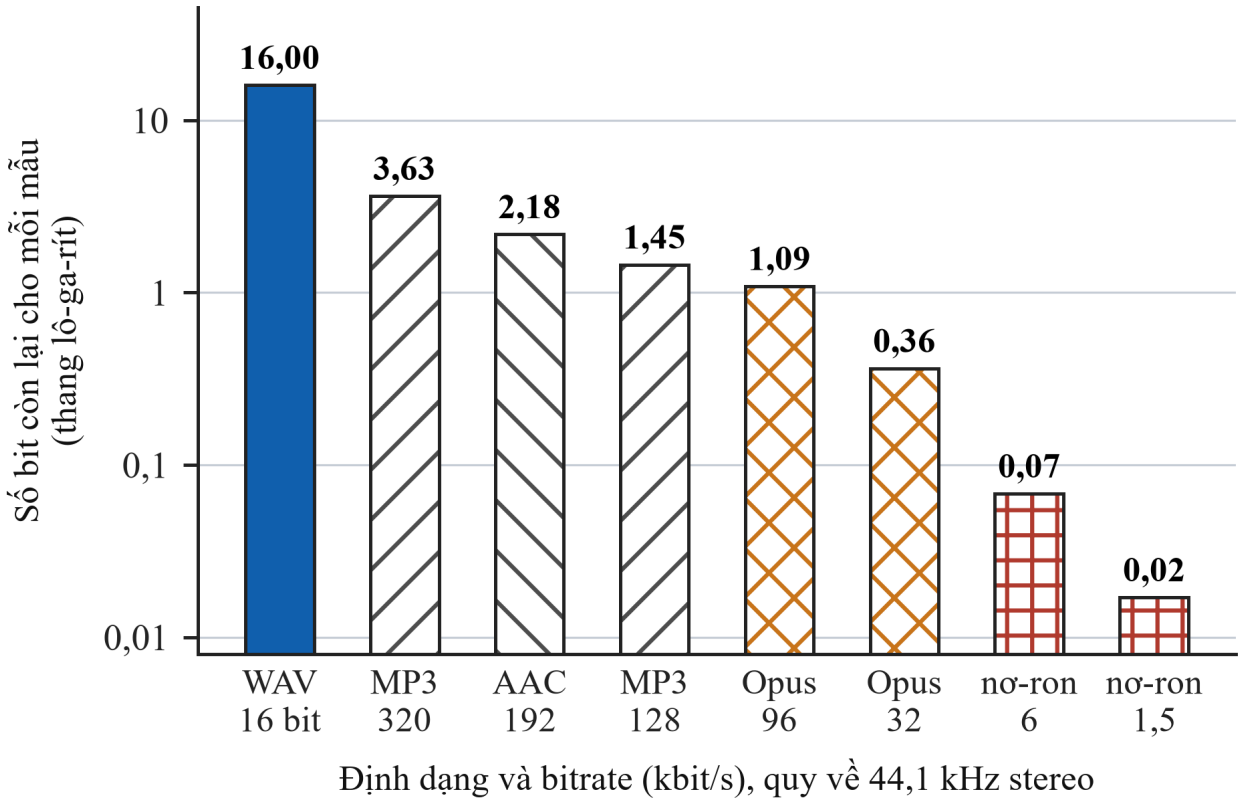
$$R = f_s \cdot b \cdot N_{\text{ch}} \quad [\text{bit/s}]\,; \qquad b_{\text{mẫu}} = \frac{R_{\text{coded}}}{f_s \cdot N_{\text{ch}}}$$

(3.10)

- R là bitrate không nén; Rcoded là bitrate của tệp đã nén.
- b_mẫu cho biết mỗi mẫu còn bao nhiêu bit sau nén.
- Ổ 64 GB chứa 101 giờ ở dạng CD, 1.111 giờ ở MP3 128 kbit/s: lệch hơn mười một lần.

Từ WAV không nén tới codec nơ-ron

- MP3 128 kbit/s chỉ còn 1,45 bit mỗi mẫu, so với 16 bit của bản gốc.
- Nhỏ hơn mười một lần mà phần lớn người nghe không phân biệt được.
- Khả thi vì phần bị bỏ đi là phần tai không dùng tới.



Hình 3.25 — Số bit còn lại cho mỗi mẫu, từ WAV không nén tới codec nơ-ron

Ba nhóm định dạng

Nhóm	Ví dụ	Giảm bao nhiêu	Khôi phục được?	Dùng khi nào
Không nén	WAV, AIFF	0 %	Không cần, chưa mất gì	Thu và xử lý
Nén không tổn hao	FLAC, ALAC	40 – 60 %	Có, chính xác từng mẫu	Lưu trữ lâu dài
Nén có tổn hao	MP3, AAC, Opus	khoảng 90 %	Không, mất vĩnh viễn	Phát hành, phát trực tuyến

Gói vali và tóm tắt sách

Không tổn hao là gói vali: gọn hơn nhưng mở ra vẫn đủ từng món. Có tổn hao là tóm tắt sách: ngắn hơn nhiều, giữ ý chính nhưng không dựng lại nguyên văn. Quy tắc nghề: thu không nén, lưu trữ không tổn hao, phát hành có tổn hao một lần ở bước cuối.

Vì sao nén có tổn hao giảm được tới 90 %

- Nén có tổn hao loại bỏ đúng những thành phần tai vốn không nghe thấy.
- Che lấp theo tần số là cốt lõi: âm nhỏ bị âm to che; đây là chỗ dung lượng co mạnh nhất.
- Che lấp theo thời gian: sau tiếng lớn khoảng 100 –200 ms và cả trước nó khoảng 2–5 ms.
- Ba điểm mù đều đã học ở chương 1.



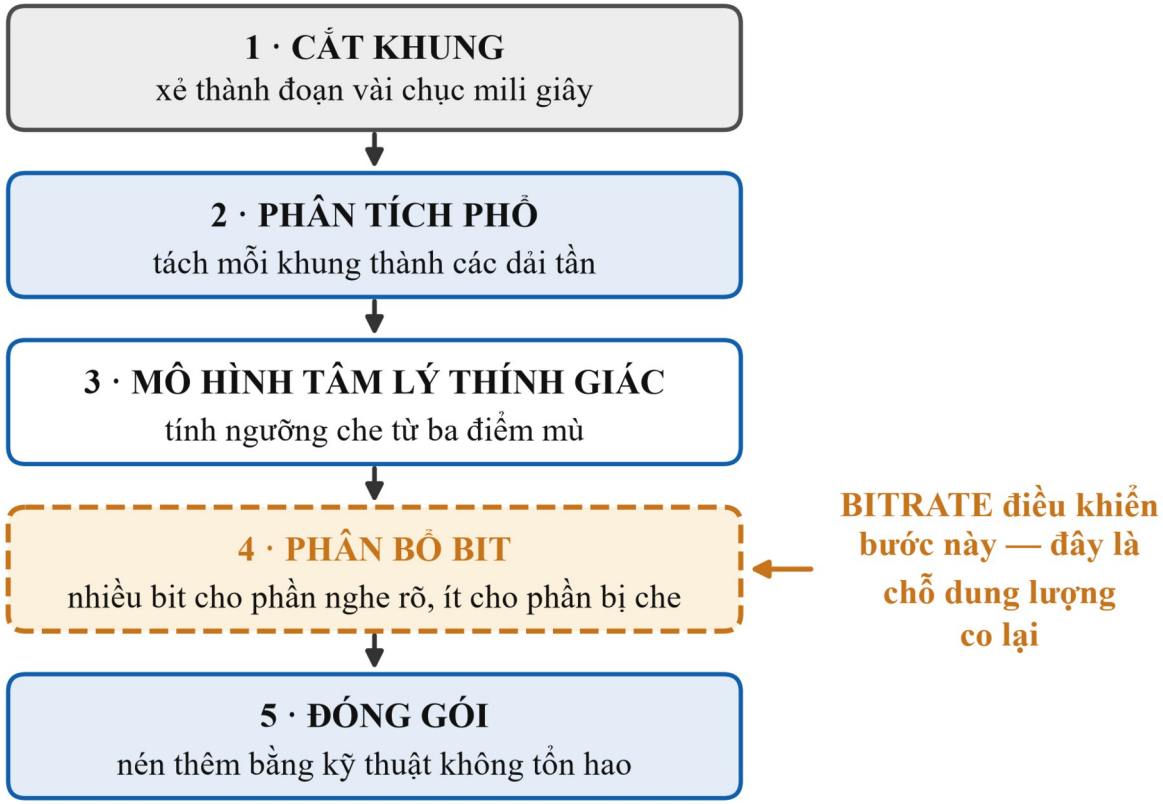
Cả ba đã học ở chương 1 như hiện tượng sinh lý.
Ở chương này chúng trở thành CÔNG CỤ KỸ THUẬT.

Nén có tổn hao không phải một thuật toán khéo — nó là kỹ thuật loại bỏ đúng những thành phần mà tai vốn không nghe thấy.

Hình 3.27 — Ba điểm mù của tai và mức tiết kiệm mà mỗi điểm mang lại

Năm bước của bộ mã hoá MP3 hoặc AAC

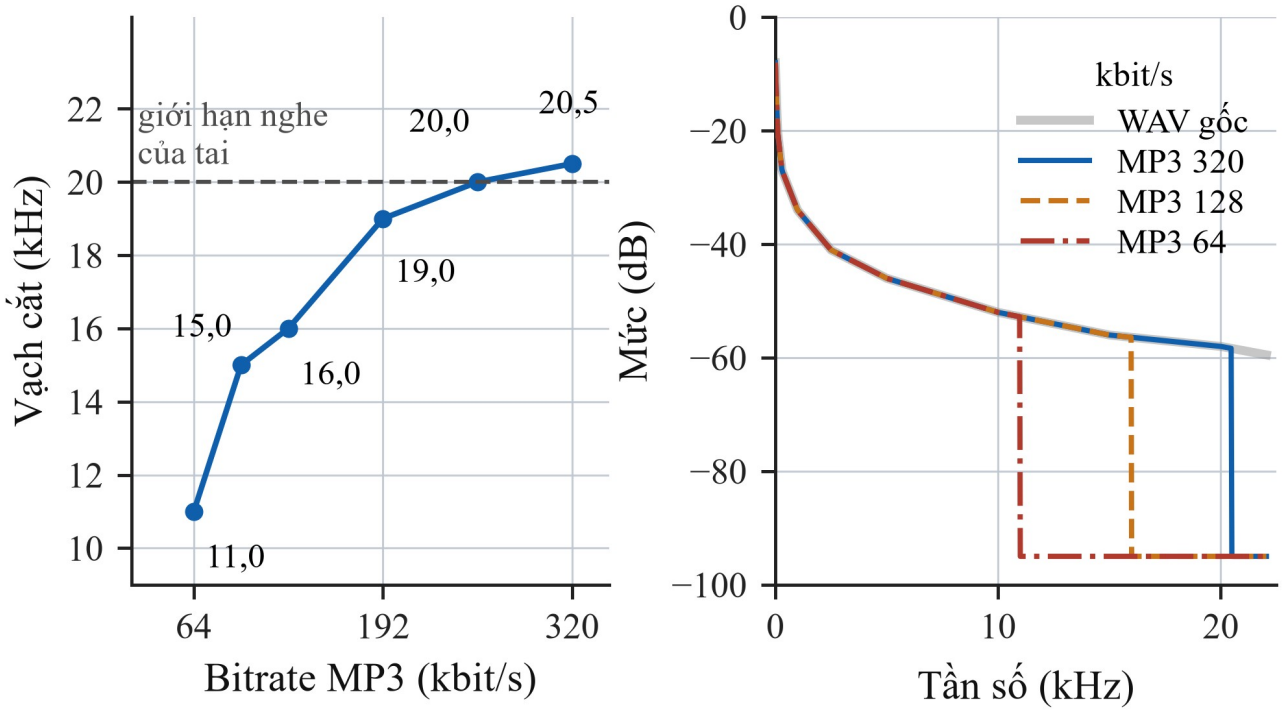
- Bước 2, phân tích ra phổ tần số, là nội dung chương 4.
- Bước 3 dùng ba điểm mù của tai.
- Bước 4 là chỗ bitrate tác động: ngân sách bit mà bộ mã hoá được tiêu mỗi giây.



Hình 3.28 — Năm bước bên trong một bộ mã hoá MP3 hoặc AAC

Vạch cắt trên phổ

- Thiếu bit thì bộ mã hoá cắt phần cao nhất trước, vì tai ít nhạy nhất ở đó.
- Trên phổ đồ hiện thành đường ranh giới ngang rất sắc, như một bức tường.
- Vị trí vạch cắt phụ thuộc bitrate.



(a) Bitrate càng thấp, vạch cắt càng tụt

(b) Vạch cắt nhìn trên phổ

Hình 3.29 — Vị trí vạch cắt theo bitrate, và hình dạng của nó trên phổ

Vạch cắt tiêu biểu theo bitrate MP3

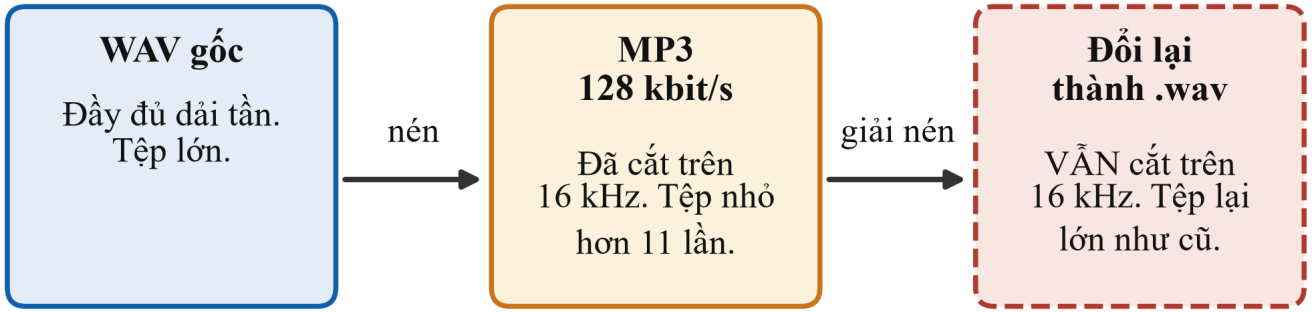
Bitrate	Vạch cắt ước lượng	Nhận xét
64 kbit/s	khoảng 11,0 kHz	nghe rõ mất dải cao, tiếng đục
128 kbit/s	khoảng 16,0 kHz	nhận ra nếu nghe kỹ bằng tai nghe tốt
192 kbit/s	khoảng 19,0 kHz	khó nhận ra với phần lớn người nghe
320 kbit/s	khoảng 20,5 kHz	gần như trong suốt

Phát hiện tệp lossless giả

Tệp đuôi .flac mà có vạch cắt sắc ở 16 kHz chắc chắn dựng lại từ MP3 128 kbit/s. Lưu ý ngoại lệ: nguồn cũ (băng từ, đĩa than) thiếu dải cao nhưng thoải dần; nén thì dựng đứng như bức tường.

Đổi đuôi tệp không phục hồi chất lượng

- MP3 đổi thành .wav: dung lượng tăng từ 0,96 MB lên 10,58 MB, gấp 11 lần, chất lượng vẫn như MP3.
- Mỗi lần nén lại là mất mát chồng lên mất mát.
- Ba quy tắc cùng dạng: gain staging (chương 2), dither (3.4), nén có tổn hao (3.6).
- Nguyên tắc: phép không hoàn tác dồn về đúng một lần, càng muộn càng tốt.



Đổi đuôi tệp không phục hồi chất lượng — chỉ đóng gói phần đã hao hụt vào một vỏ tệp to hơn.

Hơn nữa: mỗi lần nén lại là một lần mất mát chồng lên mất mát, như sao chụp lại một bản đã sao chụp.

Nguyên tắc cốt lõi: giữ bản gốc không nén để làm việc; chỉ nén có tổn hao đúng một lần, ở bước cuối cùng khi xuất bản.

Hình 3.30 — Đổi đuôi tệp không phục hồi chất lượng

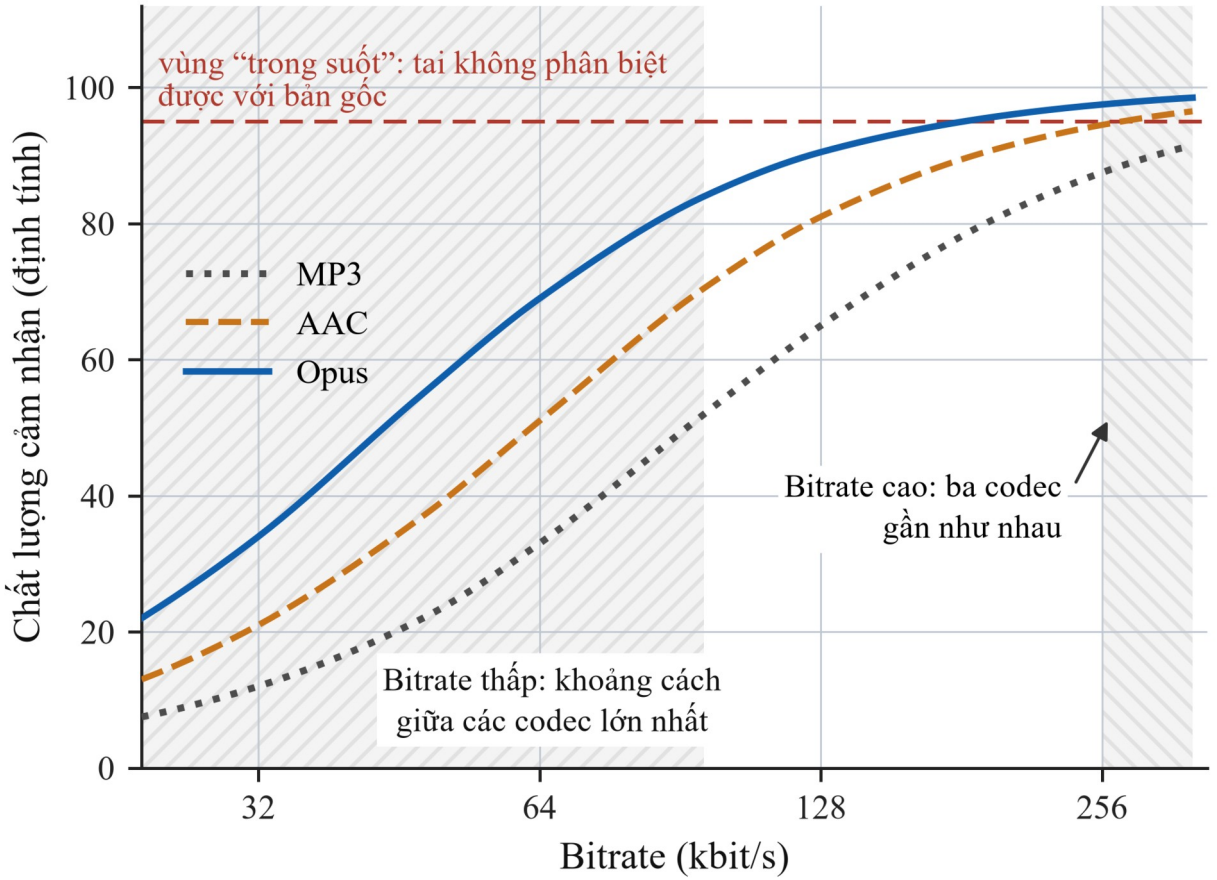
MỤC 3.7

Codec nơ-ron

- So sánh ba codec cổ điển: MP3, AAC, Opus.
- Codec nơ-ron: mạng học sâu tự học cách biểu diễn âm thanh gọn.
- Chuỗi token vừa để nén, vừa là ngôn ngữ cho AI tạo sinh.

Chất lượng cảm nhận theo bitrate

- Chênh lệch giữa các codec lớn nhất ở bitrate thấp.
- Gần như biến mất từ khoảng 256 kbit/s trở lên.
- Vì vậy MP3 dù đã cũ vẫn còn dùng được ở bitrate cao.
- Điều đáng nhớ là hình dạng đường cong, không phải thứ hạng.



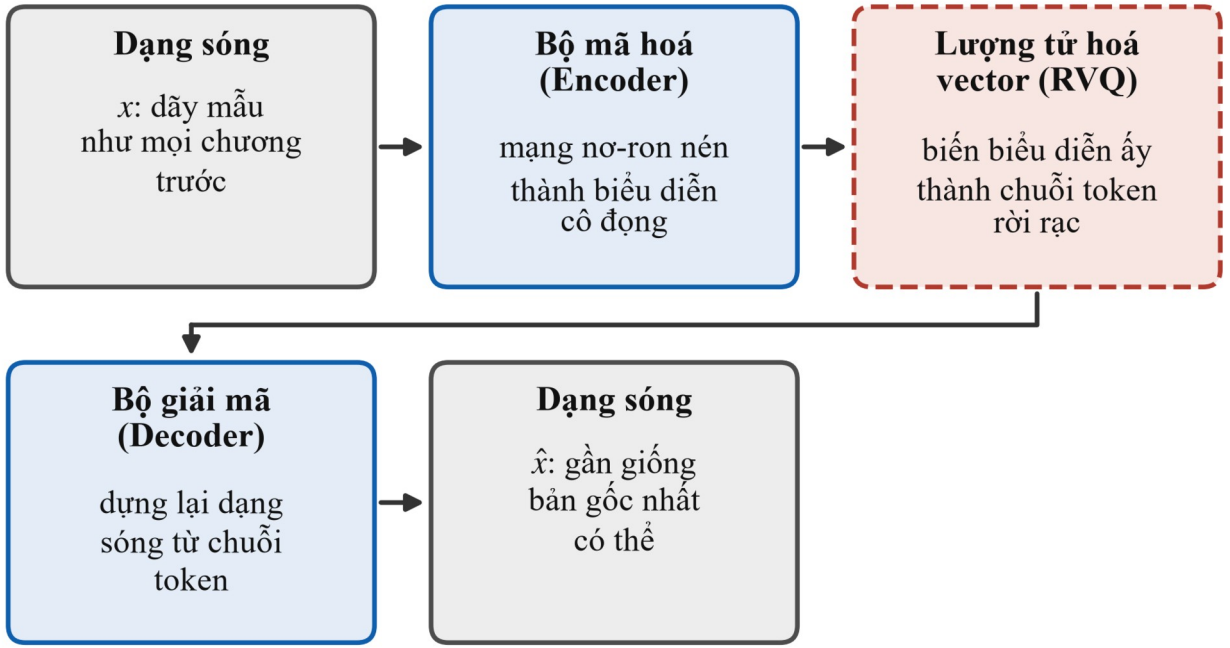
Hình 3.31 — Chất lượng cảm nhận theo bitrate — dạng định tính của ba codec

Ba codec có tổn hao phổ biến nhất

Tên codec	Điểm mạnh	Điểm yếu	Dùng ở đâu
MP3	Tương thích rộng nhất	Kém hẳn ở bitrate thấp	Kho nhạc cũ, thiết bị đời trước
AAC	Hiệu quả hơn MP3 ở cùng bitrate	Cần bản quyền trong một số ngữ cảnh	Video, phát trực tuyến, thiết bị Apple
Opus	Mạnh ở bitrate thấp; độ trễ rất thấp; mở hoàn toàn	Ít thiết bị phần cứng cũ hỗ trợ	Gọi thoại thời gian thực, nền web

Đổi từ điển cố định lấy từ điển tự học

- Codec cổ điển dựa trên mô hình tâm lý thính giác do người thiết kế.
- Từ khoảng năm 2021: mạng nơ-ron tự học biểu diễn âm thanh từ dữ liệu.
- Ba khối: mã hoá, lượng tử hoá vector, giải mã.
- Cùng ý tưởng với lượng tử hoá ở 3.4: dùng tập hữu hạn ký hiệu, nhưng là từ điển token tự học.



Điểm mấu chốt: CÙNG ý tưởng "biểu diễn gọn lại" như lượng tử hoá ở mục 3.4 — nhưng thay các mức biên độ cố định bằng một từ điển token do mạng tự học từ dữ liệu.

Hình 3.32 — Ba khối của một codec nơ-ron: mã hoá, lượng tử hoá vector, giải mã

Chuỗi biến đổi của codec nơ-ron

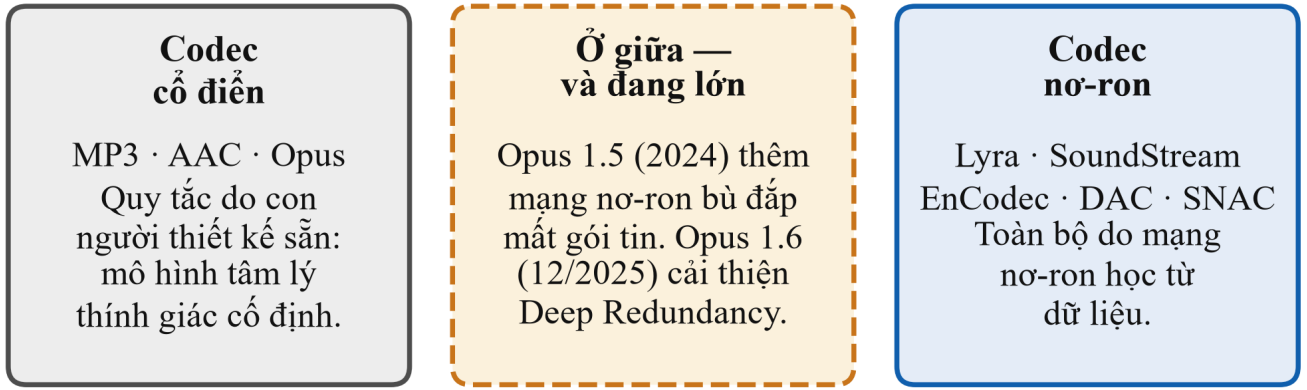
$$x \longrightarrow \mathrm{Enc} \longrightarrow z \longrightarrow \mathrm{RVQ} \longrightarrow [q_1, \ldots, q_K] \longrightarrow \mathrm{Dec} \longrightarrow \hat{x}$$

(3.11)

- Enc biến tín hiệu x thành biểu diễn z; RVQ (Residual Vector Quantization – lượng tử hoá vector nhiều tầng) lượng tử hoá nhiều tầng z thành chuỗi token q1…qK.
- Dec giải mã chuỗi token thành tín hiệu dựng lại x̂.
- Token không cách đều nhau mà tập trung vào những dạng âm hay gặp.

Opus: codec cổ điển có thành phần nơ-ron

- Opus 1.5 (tháng 3/2024): mạng nơ-ron bù mất gói tin và Deep Redundancy.
- Opus 1.6 (tháng 12/2025): cải thiện Deep Redundancy, mở rộng băng thông, thử nghiệm 96 kHz.
- Bản thảo Opus HD (tháng 10/2025): tới 96 kHz, nhưng là xử lý tín hiệu cổ điển.
- Mốc thời gian tính tới tháng 9 năm 2026.



Deep Redundancy của Opus là một trong những nỗ lực đầu tiên chuẩn hoá thuật toán học sâu tại IETF.

Đừng học thuộc "hai phe". Điều đáng nhớ là một TRỤC chung: bao nhiêu phần do người thiết kế, bao nhiêu phần do máy học.

Và một câu hỏi dùng được với mọi codec sẽ xuất hiện trong tương lai: nó đánh đổi cái gì lấy cái gì, và ở bitrate nào thì đánh đổi ấy có lợi?

Mốc thời gian ghi trên hình tính tới tháng 9 năm 2026.

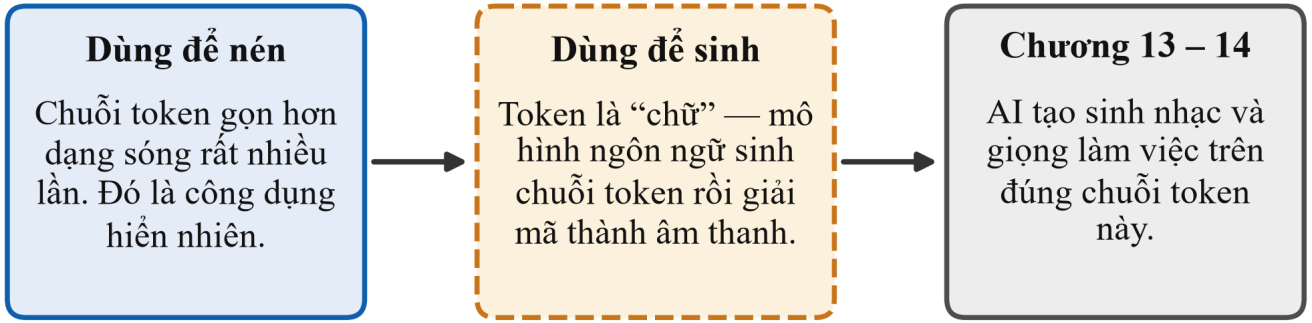
Hình 3.33 — Ranh giới giữa codec cổ điển và codec nơ-ron đang mờ đi

Không học thuộc hai phe; nhớ một trục

Trong một codec, bao nhiêu phần do con người thiết kế và bao nhiêu do máy học từ dữ liệu? Và với mọi codec mới: nó đánh đổi cái gì lấy cái gì, ở bitrate nào thì có lợi?

Chuỗi token: vừa để nén, vừa là ngôn ngữ

- Mô hình AI tạo sinh nhạc và giọng làm việc trên chuỗi token này.
- Mô hình sinh chuỗi token, bộ giải mã dựng thành âm thanh (chương 13 và 14).
- Nén và sinh cùng trả lời một câu hỏi: biểu diễn âm thanh thật gọn thì thế nào.



Hiểu nguyên lý nén hôm nay chính là bước chuẩn bị cho phần AI tạo sinh ở cuối học phần.

Không phải trùng hợp: cả hai đều trả lời cùng một câu hỏi — biểu diễn âm thanh thật gọn thì biểu diễn thế nào.

Hình 3.34 — Chuỗi token: vừa để nén, vừa là ngôn ngữ cho AI tạo sinh

Năm câu hỏi dùng được với mọi codec

- **Bitrate nào?** So ở đúng bitrate ấy chưa?
- **Đo chất lượng thế nào?** Nghe có người thật, hay chỉ số tính máy?
- **Tốn bao nhiêu tính toán, ở đâu?** Máy chủ khác điện thoại hoặc thiết bị nhúng.
- **Độ trễ bao nhiêu?** Nghe nhạc chịu được vài trăm ms; gọi thoại thì không (mục 2.6.2).
- **Ai dùng được nó?** Phần cứng, phần mềm, bản quyền quyết định khả năng thay MP3.

MỤC 3.8

Hướng dẫn thực hành

- 3.8.1 Tái hiện chồng phổ (chong_pho.py).
- 3.8.2 Hạ độ sâu bit có và không có dither (ha_bit.py).
- 3.8.3 Nhận diện tệp lossless giả bằng Audacity hoặc Spek.

chong_pho.py: sinh hai sóng, lấy mẫu thưa, so phổ

```
import numpy as np
import soundfile as sf

SR_CAO, SR_THAP = 96000, 8000
DUR = 1.0

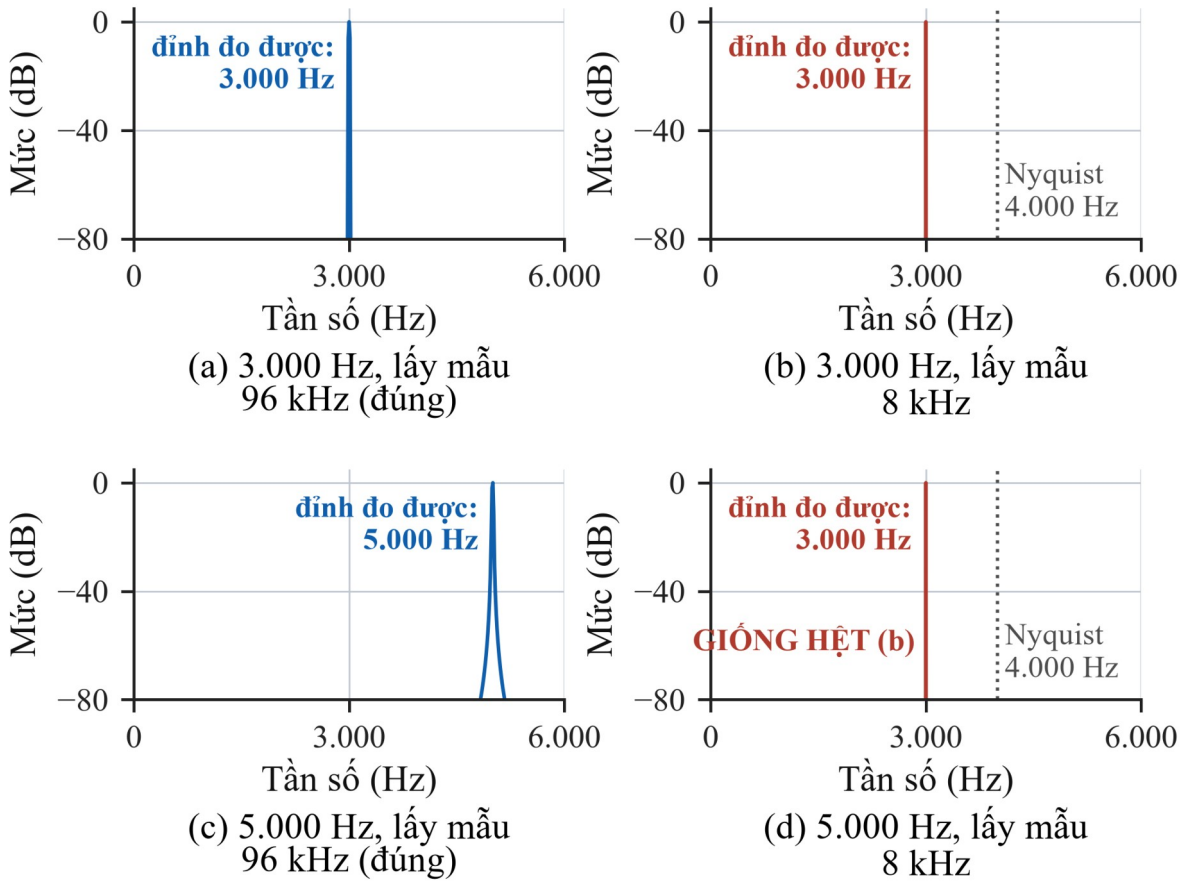
def sin_o(f, sr):
    t = np.arange(int(DUR * sr)) / sr
    return 0.5 * np.sin(2 * np.pi * f * t)

def dinh_pho(x, sr):
    w = np.hanning(len(x))
    X = np.abs(np.fft.rfft(x * w))
    f = np.fft.rfftfreq(len(x), 1 / sr)
    return f[np.argmax(X)]

for f0 in (3000.0, 5000.0):
    cao  = sin_o(f0, SR_CAO)     # lấy mẫu dày – đúng
    thap = sin_o(f0, SR_THAP)    # lấy mẫu thưa – bị gập
    print(f"{f0:6.0f} Hz | 96 kHz: {dinh_pho(cao, SR_CAO):6.0f} Hz"
          f" | 8 kHz: {dinh_pho(thap, SR_THAP):6.0f} Hz")
    sf.write(f"thap_{int(f0)}.wav", thap, SR_THAP)
```

Hai sóng khác nhau cho ra phổ giống hệt nhau

- Nghe thap_3000.wav và thap_5000.wav: nghe giống nhau.
- Mở bằng Audacity, chế độ Spectrogram: cùng một vạch ở 3.000 Hz.
- Không phải hạn chế phần mềm: hai tệp chứa dãy số giống hệt nhau.
- Độ phân giải phép đo là fs/N; cần nhân cửa sổ Hanning (chương 4).



Hình 3.35 — Kết quả phải thấy: hai sóng khác nhau cho ra phổ giống hệt nhau

ha_bit.py: lượng tử hoá một đoạn nhỏ, hai cách

```
import numpy as np
import soundfile as sf

SR, DUR, BIT = 44100, 3.0, 8
t = np.arange(int(DUR * SR)) / SR

# đoạn NHỎ — khoảng −34 dBFS, đúng chỗ dither quan trọng
x = 0.02 * (np.sin(2 * np.pi * 220 * t)
            + 0.5 * np.sin(2 * np.pi * 660 * t))

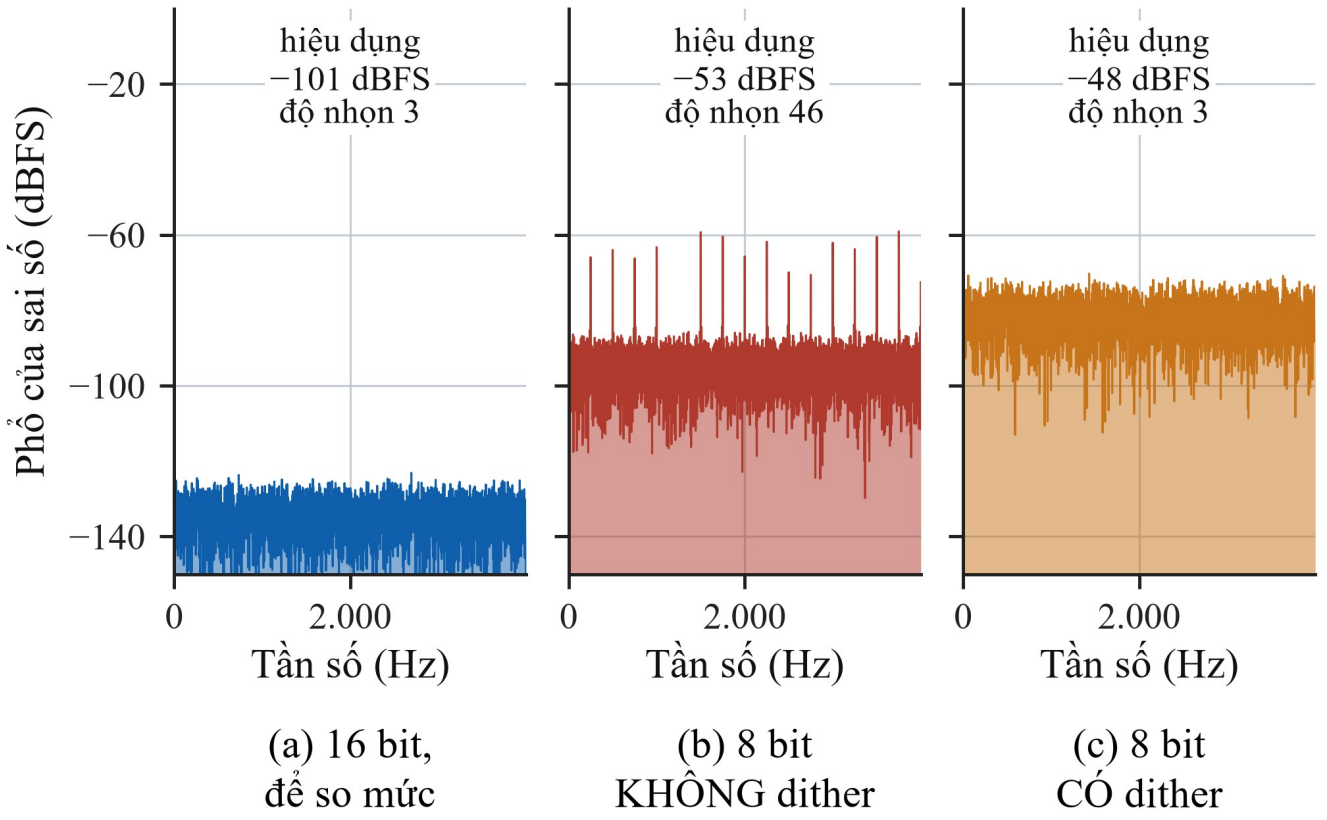
buoc = 2.0 / (2 ** BIT)          # bước lượng tử
rng  = np.random.default_rng(0)
tpdf = buoc * (rng.random(len(x)) - rng.random(len(x)))

khong = np.round(x / buoc) * buoc            # không dither
co    = np.round((x + tpdf) / buoc) * buoc   # có dither

for ten, y in (("khong_dither", khong), ("co_dither", co)):
    e = y - x
    print(ten, "| sai số hiệu dụng =",
          round(20 * np.log10(max(np.sqrt(np.mean(e ** 2)), 1e-12)), 1),
          "dBFS")
    sf.write(f"{ten}.wav", y, SR)
```

Phổ của riêng sai số ở ba trường hợp

- Nghe hai tệp bằng tai nghe, tăng âm lượng.
- Không dither: tiếng lạo xạo bám theo giai điệu (méo).
- Có dither: nhiễu nền đều, to hơn một chút nhưng dễ chịu hơn nhiều.
- Đó là sự khác nhau giữa méo và nhiễu.

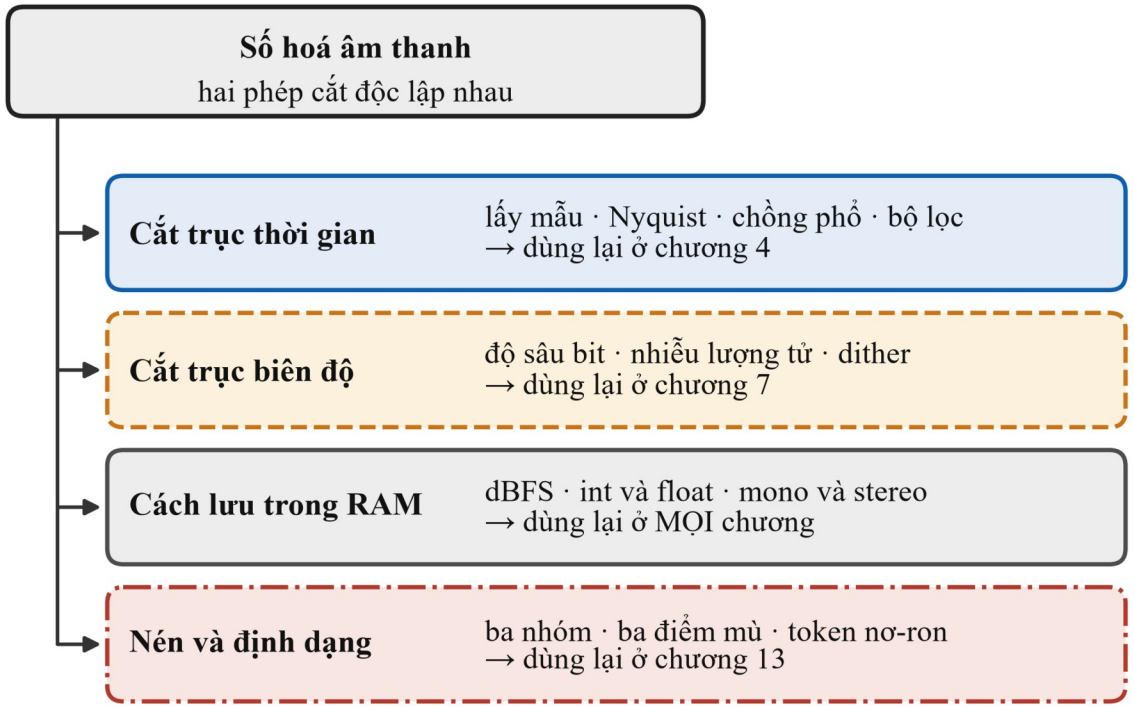


Hình 3.36 — Phổ của riêng sai số ở ba trường hợp, trên cùng một thang tuyệt đối

Bảy bài tập tính toán và hai tình huống

- Bài 1–2: Nyquist, chồng phổ, chọn tần số lấy mẫu (ví dụ tiếng dơi tới 45 kHz).
- Bài 3–4: độ sâu bit, dải động, dBFS và headroom.
- Bài 5–6: dung lượng, bitrate, bit cho mỗi mẫu của Opus.
- Bài 7: chọn cấu hình thu, xử lý, phát hành cho một dự án phim tài liệu.
- Tình huống 1: kiểm tra bộ sưu tập lossless 24 bit/96 kHz; tình huống 2: vạch lạ ở 3–5 kHz.
- Yêu cầu trình bày phép tính; đáp số in sẵn trong chương.

Bốn nhánh khái niệm của chương và chương sẽ dùng lại



Nhánh thứ ba là nhánh dùng nhiều nhất: từ chương 4 trở đi, mọi dòng mã đều làm việc trên dãy số ấy.

Hình 3.37 — Bốn nhánh khái niệm của chương 3 và chương nào sẽ dùng lại từng nhánh

Những điều cần nhớ

1. Số hoá gồm hai phép độc lập: lấy mẫu cắt trục thời gian, lượng tử hoá cắt trục biên độ.
2. Nyquist: fs > 2·fmax; dựng lại bằng sinc. Chồng phổ biến thứ vô hại thành có hại, phải chặn bằng bộ lọc từ trước.
3. Mỗi bit thêm 6,02 dB dải động; dither đổi méo lấy nhiễu, dùng đúng một lần ở bước hạ bit cuối.
4. Nguyên tắc cốt lõi: đọc vào đổi sang số thực, xử lý ở số thực, chỉ hạ về số nguyên ở bước ghi cuối.
5. Nén có tổn hao dựa trên ba điểm mù của tai và không hoàn tác được: nén đúng một lần, ở bước cuối.
6. Codec nơ-ron thay từ điển cố định bằng từ điển token tự học, nối nén với AI tạo sinh.